

ASM G03

The POLY 88
Editor/Assembler

PolyMorphic
Systems

460 Ward Drive Santa Barbara California 93111 (805) 967-2351

This manual is PolyMorphic Systems part number 310110

Copyright 1977, Interactive Products Corporation.

GØ3

1. Introduction to the POLY 88 Editor/Assembler

The POLY 88 Editor/Assembler is intended to be used mostly for creating and assembling assembly language programs. It includes a wide range of editing features, allowing you to create and edit text. And it also includes a flexible assembler and printer driver.

1.1 Machine Language.

Although we often talk of various "computer languages," all computers, including the POLY 88, really "understand" only one language, and that is "machine language" or binary code. All the operations of a computer involve the switching of thousands of switches or bi-stable devices. Since each switch has only two possible states, the only "language" a computer can really use is one that sets its switches to one or the other of their two possible states; thus all the terms of machine language must be built up of two values, such as 0 and 1. A computer accepts and understands instructions in the form of groups of 0s and 1s.

It is possible to write programs in machine language, but few people do. Though machine language is perfectly suited to the machine, it is an awkward instrument for the program writer. It must build its statements from 0s and 1s, and the meanings of these binary statements are hard for the programmer to remember. So programs are usually written in various "higher" languages. Then the computer itself is instructed to interpret or translate the higher-level language program into machine language.

1.2 Assembly language.

Assembly language is the simplest and most direct of all computer languages above the level of machine language. In the case of the POLY 88, for instance, the computer's central processor can perform seventy-eight different operations, each one of which has its corresponding binary expression or opcode. In assembly language, the opcode is replaced by a mnemonic or word-like term that indicates the nature of the operation-- like MOV or JMP-- and so is easier to remember and use than the opcode. So a program written in assembly language is identical to a program written in machine language, except that the instructions to the computer's central processor are expressed in word-like terms instead of in binary. Despite the resemblance, the computer cannot understand the assembly-language program until the mnemonics are "assembled" or translated into machine language. That translation is the job of the assembler.

1.3 Assemblers

Assemblers are themselves machine-language programs that have the special function of translating assembly-language programs into machine language. When you write a program in assembly language, you turn the finished program over to the assembler for translation. A good assembler should include an "editor" or software that allows you to create your program by typing on the computer keyboard and reading the video screen. The editor should allow you not only to create text but also to change it quickly and easily as you go along. When you are done writing, the assembler should be able to assemble and "list" or show you your program immediately, listing on any desired output device (e.g. video screen or printer) and indicating errors if any in your assembly-language program. The POLY 88 Editor/Assembler includes all these features.

1.4 BASIC.

Programmers often use higher-level languages than assembly language. A frequent choice of micro-computer users is BASIC. It is in a way even easier to use than assembly language, because it can summarize many machine operations in a single term. But it loses the one-to-one program-instruction-to-machine-operation relationship that assembly language retains. And assembly-language programs are frequently somewhat more efficient than BASIC programs in that they will cause the processor to perform fewer operations in solving a given problem. It is not unusual for a machine-language program to do a given job ten or a hundred times faster than its BASIC counterpart.

PolyMorphic Systems markets a useful version of BASIC for the POLY 88 user. We might note here that when you use BASIC, you are using a machine-language program that can interpret BASIC statements into machine language. BASIC software does not translate your program into machine language, creating as a result a machine-language program, as does the Editor/Assembler. Instead, it intervenes between the program written in BASIC and the central processor, interpreting BASIC into machine language for the benefit of the processor. For that reason, when you run a program written in BASIC, you must have BASIC itself currently stored in machine memory (RAM) at the same time. An assembler converts your assembly-language program into a machine-language program; the machine-language program will then run directly without any help, leaving the rest of RAM free for other uses--another advantage of assembly-language programming.

1.5 Other languages.

Programmers who work with large computers use other higher computer languages. These languages too may produce less efficient programs than their assembly-language equivalents. But when doing extensive computations on big machines, ease of program-writing is more important than efficiency. Two familiar high-level languages are FORTRAN, created for doing scientific computations, and COBOL, which plays the same role in business.

2. Using the POLY 88 Editor/Assembler.

2.1 Loading.

Take a minute to load the Editor/Assembler according to the instructions below, and follow the directions indicated in the following discussion.

To load the POLY 88 Editor/Assembler, you will of course need the cassette interface and a cassette recorder. Insert the Editor/Assembler tape into the recorder, set the volume control to nine, and reset your POLY 88. Now type either P or B for Poly-Phase or Byte, whichever loading mode your system and tape require. (You will get an automatic carriage return after typing P or B.) Now type ASM and a carriage return, and start your tape. When the Editor/Assembler is fully loaded, you will get the assembler prompt *.

2.2 Writing and editing.

To create an assembly-language program, go directly to the edit mode. Do this by typing EDIT and a carriage return. (From this point we will assume that you recall that all commands end with a carriage return.) The editor prompt, a solid rectangular cursor, will appear. The cursor indicates your exact present location in the text; in other words, when you type in a character, it will appear in the cursor location, and the cursor will move one space forward (to the right or to the beginning of the next line).

2.2.1 Adding and deleting text.

Create text just as you would on a typewriter. Characters appear as you type them in. To delete, hit DELETE (or RUB OUT). The character to the left of the cursor disappears, and the cursor moves to the left one space. If the cursor is currently located at the left end of a line, the carriage return at the right end of the previous line is deleted, and the cursor moves up to that position.

You can delete one entire word by hitting CTRL-W (hold down the CTRL key and type W), and delete one whole line by hitting CTRL-X. Again, deletion is to the left.

Note, by the way, that a line on the screen is not necessarily a true line of text. Screen lines notwithstanding, a true line is the text between two carriage returns. If you type to the end of a line on the screen, your text will automatically "wrap around" and resume on the next line, even if you don't type in a carriage return. If you then list your text on a printer, your apparent two lines of text will print as a single line. Furthermore, the assembler will not get the carriage return it needs to delimit program lines, and your program will be assembled incorrectly.

2.2.2 Cursor controls.

There are several ways to move the cursor to the point in your text where you want to add or delete. Move the cursor one character to the right by hitting the right-pointing arrow key (CTRL-S on old keyboards without arrow keys), to the left one character by hitting the left arrow key (CTRL-T on old keyboards). Move it up one line by hitting the up arrow (CTRL-Q) and down one line by hitting the down arrow (CTRL-R). These commands always leave the cursor at the beginning of a line. You will quickly learn to move around in text with ease; for instance, to move from the beginning to the end of a line, hit the down arrow, then the left arrow.

2.2.3 Paging; End-of-Text; Beginning-of-Text.

You can move more rapidly within your text by using other commands. CTRL-P shows you the previous "page" or screenful (fifteen lines) of your text; CTRL-N shows you the next fifteen lines. CTRL-B takes you to the beginning of the text currently in memory and CTRL-E to the end.

2.3.4 Search Strings

To find the occurrence of a particular word or other string of characters in your text, use the CTRL-F feature. Hit CTRL-F; you will see a double cursor. Now type in the word or other string you want to find in your text. It will appear between the cursors. **Control characters (like carriage return) will appear as Greek letters.** Then hit ESC (the escape key-- on old keyboards without ESC, hold down CTRL and SHIFT and hit K). The screen will display the location of that word or other string in your text, with the cursor just to the right of the found string. To find the next and all subsequent occurrences of that string, hit CTRL-C ("Continue search").

Note that CTRL-F always searches forward in the text currently in memory, so as a rule use CTRL-B before using CTRL-F to insure that you are finding the earliest occurrence of the sought string. If the string you are searching for does not occur in the text, the search string and extra cursor will simply disappear.

2.2.5 Help

One last note about the editing commands: hit CTRL-H (Help) to see a brief summary of the commands. Return to your text by hitting any key. (BACKSPACE is the same as CTRL-H.)

2.2.6 Sample program.

Here's a look at how you format an assembly-language program when using the POLY 88 Editor/Assembler. Text falls into four columns or "fields": label, assembly mnemonic, parameter, and comment. (The assembler ignores comments; they appear as text in a listing.) The assembler recognizes the end of a field when it encounters a space or a tab, so move from one field to another by spacing or hitting a tab (CTRL-I on old keyboards). Obviously, since the assembler ignores comments, it also ignores spaces and tabs in comments.

After seeing a carriage return, the assembler anticipates a label. If it encounters a space or a tab instead, it then knows that the next thing it encounters will be an assembly mnemonic. And so forth. Comments must always be preceded by a semi-colon. Hex numbers beginning with a letter must be preceded by a zero.

Always start with a comment line identifying the program; that line will come up on the screen whenever you load the program.

```
;THIS PROGRAM GENERATES THE CHARACTER SET ON THE VIDEO SCREEN
;
;      ORG      0D00H      ;WHERE WE ASSEMBLE THE CODE TO RUN
;
;EQUATES
;
SCRST EQU      0F800H      ;LOCATION OF THE VIDEO RAM
SCREND EQU     0FCH        ;SCREEN END COMPARE ADDRESS
;
;THE PROGRAM
;
      LXI      H,SCRST    ;POINT HL AT BEGINNING OF SCREEN
LOOP  MOV      M,L        ;WRITE LSB OF ADDRESS TO SCREEN
      INX      H          ;POINT TO NEXT LOCATION
      MOV      A,H        ;ARE WE AT THE END OF THE SCREEN?
      CPI      SCREND
      JNZ      LOOP       ;IF NOT AT END GO BACK AND WRITE MORE
HLTAGN HLT            ;WE'RE DONE SO STOP
      JMP      HLTAGN     ;WHERE TO GO AFTER INTERRUPT OCCURS
      END                ;THAT'S ALL FOLKS!
```

2.2.7 POLY 88 assembly pseudo-ops.

A few assembly mnemonics do not get assembled into machine-language opcodes. These are the "pseudo-ops," which instruct the assembler to perform certain assembly functions. The complete set of assembly mnemonics consists of the instruction set for a given central processor plus the pseudo-ops of a given assembler. The instruction set for the POLY 88's Intel 8080A central processor is fully described in the POLY 88 CPU Card manual. The assembler pseudo-ops are described below.

a) DB "Define byte."

The mnemonic DB causes the parameter that follows it to be evaluated and put into the machine language file in the form of an eight-bit byte. More exactly, it evaluates the parameter as a

16-bit quantity, drops the high eight bits, and puts the remaining eight-bit byte into the machine language file. The high eight bits can be non-zero; they will still be ignored (a V error will be displayed, but you can ignore this). Evaluation of a letter character results in the ASCII code for that character being entered into the machine-language file. Use the DB pseudo-op in the following format:

Label	Mnemonic	Parameter	Comment
	DB	'ABC...'	;PUT TEXT ON SCREEN
	DB	'HELLO', 63	;MIX KINDS OF CHARS.
	DB	80H, 146	;MIX HEX AND DECIMAL

b) DW "Define word."

The pseudo-op DW evaluates the parameter as a 16-bit quantity, but does not drop the more significant byte. Instead, it puts both bytes into the machine-language file, in byte-reversed form (less significant byte first).

c) DS "Define storage."

The DS pseudo-op evaluates the parameter and allocates that number of memory locations (i.e., it advances the assembler's location counter that number of addresses).

Label	Mnemonic	Parameter	Comment
	DS	100	;ALLOCATE 100 BYTES

d) EQU "Equate."

The EQU pseudo-op serves to equate a label with a value or other expression.

Label	Mnemonic	Parameter	Comment
VT	EQU	0BH	;VT NOW ASSUMED TO = 0BH
LOOP1	EQU	LOOP2	

e) ORG "Origin."

This sets the assembler's location counter to the address expressed in the parameter.

Label	Mnemonic	Parameter	Comment
	ORG	2000H	;START OF USER RAM

More than one ORG may be used per program.

f) END

The END pseudo-op terminates the input of text to the assembler. If the assembler has now completed the first of its two passes through the assembly-language program, creating a table of

labels, it goes on to its second pass. If it has now completed its second pass, it displays the number of assembly errors and returns to the assembler prompt * .

9.

LON/LOFF/ENDS

2.3 Assembling.

Having written your program in edit mode, you will want to switch to assembly mode and assemble it. First you must make a couple of decisions.

2.3.1 Assembly addresses.

As the assembly process takes place, the assembled program gets stored into memory. You must decide where in memory you want them stored. And each assembled program must be tagged with an address that determines where in memory the program will run when it is used. These two addresses are sometimes, but not always, the same. For instance, the assembler itself resides in the memory space starting at address 2000H (2000 hexadecimal). You may want to run your program within the space currently occupied by the assembler, which is perfectly possible since the assembler need not be in memory when you run your program. But obviously when you assemble your program, the assembler is there, in the way. So you may want to choose two different addresses: an origin, which determines where in memory the program will run when used; and an assembly address, which determines where in memory the assembly process will occur.

When you command the computer to assemble your program, you will give it a command that looks something like this:

```
ASMB 1234 5678
```

in which the first number is the origin address and the second the assembly address. If you do not state an assembly address, the assembler assumes that the two addresses are the same.

You must make one other decision before assembling: how you want to see your program listed.

2.3.2 Listing options.

As your program is assembled, it will be listed on the video screen and (if you wish) on a printer. The assembler automatically identifies errors to be corrected. You can list the program in several different ways by using the following commands (addresses deleted here):

ASMB This command assembles the program and lists it in its entirety, with errors flagged.

ASMBE This command also causes the program to be assembled, but lists errors only. You will probably want to use this command the first time you attempt to assemble your program. Check the errors and return to EDIT to correct them.

ASMBs This command tells the assembler to use the same symbol

table it used previously. Use this when assembling the second half of a lengthy program.

Finally, the E and S features can be used together: ASMBES.

2.3.3 Assembling.

Having made the above decisions, you are ready to assemble. To go from the edit mode to the assembly mode, hit ESC (ALT MODE on POLY 88 keyboards without an ESC key). Note: ESC (or ALT MODE) always returns you to assembler mode. When you leave the editor, you get an advice of the number of bytes free (available for use) in memory, and the highest free address. When you see the assembler prompt *, give one of the ASMB commands with addresses as required to start assembly. Assembly will begin, and the program will be listed in assembled form on the video monitor.

You can list on a printer (i.e. output to an RS-232 port) by preceding the ASMB command with the command HARD, followed by a carriage return. To disable the printer and list on the screen only, precede ASMB with the command SOFT. The video display will fill, then stop, displaying one screenful; after you have read it, hit any key for the next screenful.

Your fully assembled program will be listed like the example below, which is the assembled version of the sample program above. Note that on the left, two new columns have been added. First is an address, expressed in hexadecimal. This is the address at which each assembled machine-language instruction will be stored when the program is run. (To be exact, it is the FIRST address of each instruction, since an instruction is often more than one byte.) The first few lines of this program are not in fact program instructions, so nothing appears in the far left column in those lines. In the next column-- the column that contains 0D00, F800, 00FC, etc.-- is the machine-language translation of the instruction in that line, or the value assigned to a label by an EQU pseudo-op. Note that these values, whether they are the machine-language equivalent of an instruction or the equivalent of a label, are expressed in hexadecimal, NOT in binary. As we said before, binary is an awkward medium for the programmer, so the assembler lists machine language in hexadecimal. Since two hex characters equals one eight-bit byte, every two hex characters represents a byte of machine-language instruction. Look, for instance, at the first program instruction, LXI. This mnemonic translates into a three-byte machine statement. A look at the second column from the left shows that the machine-language equivalent of the assembly-language term LXI, when expressed in hex, is 2100F8. Or, dividing it into three bytes, 21, 00, F8. Now, since LXI is a three-byte instruction, and since the first column lists the starting address of each instruction, the second address in that column should equal the first-address plus three. And it does-- 0D00 and 0D03.


```

;THIS PROGRAM GENERATES THE CHARACTER SET ON THE VIDEO-SCREEN
;
0D00          ORG      0D00H ;WHERE WE ASSEMBLE THE CODE TO RUN
;
;EQUATES
;
F800  SCRST  EQU      0F800H ;LOCATION OF THE VIDEO RAM
00FC  SCREND EQU      0FCH   ;SCREEN END COMPARE ADDRESS
;
;THE PROGRAM
;
0D00 2100F8          LXI      H,SCRST ;POINT HL AT BEGINNING OF SCREEN
0D03 75             LOOP    MOV      M,L ;WRITE LSB OF ADDRESS TO SCREEN
0D04 23             INX      H ;POINT TO NEXT LOCATION
0D05 7C             MOV      A,H ;ARE WE AT THE END OF THE SCREEN?
0D06 FEFC           CPI      SCREND
0D08 C2030D          JNZ      LOOP ;IF NOT AT END GO BACK AND WRITE MORE
0D0B 76             HLTAGN  HLT      ;WE'RE DONE SO STOP
0D0C C30B0D          JMP      HLTAGN ;WHERE TO GO AFTER INTERRUPT OCCURS
                                END    ;THAT'S ALL FOLKS!

```

If there had been errors in the assembly-language program, the assembler would have flagged them with error flags in the left margin. These flags are letters indicating the type of error.

R	Register error. Bad register name (see CPU Card manual).
S	Syntax error. The system doesn't understand this line.
U	You have referred to a label that you have left Undefined.
V	The system has been given a Value of more than eight bits. It wants a value of eight bits or less.
M	You have an EQU but no label. ("Missing.")
A	Argument error (bad parameter).
O	Opcode error. The system doesn't recognize the mnemonic.
L	Label error. Labels can consist of ^{uppercase} letters and numbers but no other characters.
D	Duplicate error. You tried to re-define a label.

As it turned out, our assembly-language program had no errors. If the assembler discovers errors in the assembly-language program, you will want to return to EDIT and correct them. If you don't have a hard-copy listing, first write down the errors. Then type EDIT and a carriage return and correct the errors. Then repeat the assembly.

2.4 Storing programs on tape.

You can store the assembly-language program or the machine-language program on tape by using the SAVE and DUMP commands. Use SAVE to output text (i.e. the assembly language program, called the source code) to tape, specifying Byte or PolyPhase and stating a name for the file (i.e. the program):

```
SAVEB /name/
```

```
SAVEP /name/
```

When you load that tape into the POLY 88, the first line of the program will appear on the screen as a means of identifying the program.

Output machine language (i.e. the assembled program, called the object code) with the command DUMP plus B or P and the beginning and ending addresses of the memory space you want to dump (these are called the From and To addresses):

```
DUMPB /name/ FFFF TTTT
```

```
DUMPP /name/ FFFF TTTT
```

These addresses must be expressed in hexadecimal. After you have given the DUMP command, the system will ask you for load and start addresses. Give a hexadecimal load address stating the first address in memory that you want the program to occupy whenever it is loaded, and start address stating the address of the first instruction you want executed whenever the program runs.

The DUMP command always adds an auto-execute record to the dumped program. To load a dumped program, use the "front panel bootstrap" loader-- in other words, use the same loading procedure you used to load the Editor/Assembler tape. The program will be loaded automatically, starting at the load address you stated when you dumped the program, and then will be run automatically, starting at the start address you stated when you dumped the program.

2000-2050

2.5 Summary of assembly procedure.

- a) Load the Editor/Assembler into your POLY 88 and go into EDIT to write your program. Begin with a comment line and comment fully throughout. You can create blank lines at the top of the text by putting the cursor at the top of the screen and hitting RETURN. Use blank comment lines to separate blocks of program.
- b) When you have finished your text, look it over carefully for errors or omissions. Then leave EDIT and go to the Assembler by hitting ESC. You can now SAVE your assembly-language program to tape if you wish, or assemble it. Assemble using the ASMBE command and citing the address given when you departed from the editor, or a higher address (but not so high that your program will run off the top of memory). Note any errors and return to EDIT to correct them. Continue this process till ASMBE rereveals no errors (you will get the message "No assembly errors"). Before or after editing, you can LIST your program using the appropriate commands.
- c) Having eliminated assembly errors, now choose a load address (origin) and an assembly address as described above. These can be the same if the permanent load address is higher than the Editor/Assembler (or is in on-board RAM, as in the sample program); in that case, state the address just once. Use the command ASMB and the address or addresses to produce the final assembled program. Use listing options as well. (If you give the assembler no listing option, it assumes you want the program listed on the screen only. If you have previously used the HARD command, however, you must then use the SOFT command to disable the printer.)
- d) Now you can test-run your program by using the CALL command or record it on tape using the DUMP command. If you CALL your program, it will execute. When the computer encounters a RET (return) instruction in the main program, it automatically returns to the assembler. Note that you can't test-run in this way if your program is addressed to run in the space occupied by the Editor/Assembler.

2.6 The printer driver.

So that it can list on a printer, the POLY 88 Editor/Assembler includes a block of program devoted to the operation of a printer. This part of the software comes into play automatically when you select hard copy by using the HARD command. It ensures that the output to the RS-232 port includes values that allow the printer sufficient time to execute line feeds and form feeds, etc.

The values supplied in the printer driver block are pre-defined to allow the use of a Diablo HyType printer. If you use some other printer, you will have to customize the printer driver block.

2.6.1 Pre-defined Editor/Assembler printer driver values

Here is what a listing of the printer driver block looks like, with the values it is pre-assigned in marketed form. Change these values by using the Front Panel Mode to create a customized driver.

Address	Label	Constant	Brief Explanation
2008H	PDCHR	7FH (rubout)	Pad character
2009H	PDCNT	1	Number of pad characters
200AH	NFPAD	1	Pad count for form feed
200BH	NFPDL	1	Pad count for page overflow
200CH	NLPP	3FH	Number of lines per page
200DH	FFPAD	7FH	Number of pad characters for form feed
200EH	ACRCH	0AH (LF)	Character sent after carriage return
2012H	SPEED	16H	Printer baud rate

The labels used in the printer driver have the following meanings:

PDCHR
PDCNT

When the computer outputs a carriage return instruction to the printer, output must pause while the printer executes the carriage return. This is actually accomplished by sending the printer a number of characters that it can ignore while the printer head is flying back to the left margin. These are called "pad characters." In the case of the HyType, the computer outputs one delete character. The type of character to be sent during execution is defined in the PDCHR line and is sent the number of times stated in the PDCNT line. A zero value in any count field actually means 256 counts; the minimum value is one.

NFPAD

When a form feed character occurs in the text, forcing the printer to move on to the top of the next page, the output of characters must pause till the printer has completed the form feed. The character defined in PDCHR is sent the number of times specified by NFPAD.

FFPAD
NFPDL

These values determine what character will be sent (FFPAD) how many times (NFPDL) whenever the printer is moving from the end of one page to the beginning of the next page ("page overflow"). The HyType is given one delete.

NLPP

This line defines the number of lines that will constitute a full printed page.

ACRCH

The after-carriage-return character is sent after the carriage return pad characters are sent. Some printers, including the HyType, need to be sent a line feed after a carriage return; others will execute a line feed automatically after a carriage return. Here, ACRCH defines the character to be sent after the end of the carriage return pad to be a line feed (0AH, the ASCII value in hex of a line feed). The computer then sends pad characters (namely the character previously defined in PDCHR) the number of times defined in PDCNT. This allows the printer time for execution of the line feed.

SPEED

This defines the baud rate of the printer. It is pre-defined to be 300 baud (16H), the baud rate of the HyType. Other printers accept information at different baud rates. Modify this value to match your printer's baud rate in accordance with the following table.

HEX	BAUD RATE	HEX	BAUD RATE
11H	50	19H	1200
12H	75	1AH	1800
13H	110	1BH	2400
14H	134.5	1CH	3600
15H	150	1DH	4800
16H	300	1EH	7200
17H	600	1FH	9600
18H	900		

2.6.2 Customizing the printer driver block.

First determine your printer's characteristics. Then poke the appropriate values into the printer driver block, using Front Panel Mode. Save the customized driver on tape, appending to it the remainder of the assembler with the original printer driver deleted. Carefully label this file with a name that includes the name of the printer type it is intended for. You may find that you have to experiment, tuning these values, to get your printer to run.

2.7 NSYM

Another pre-determined value sets the maximum number of symbols possible in a single program.

NSYM (Number of Symbols) is located at address 2006H. It is pre-defined to be 128 (80H). NSYM pre-allocates 128 symbol locations of ten bytes each. The "bytes free" message displayed upon loading of the assembler is the total number of bytes currently unused in memory, minus enough bytes to accommodate 128 symbols, at ten bytes per symbol. You can enlarge or reduce this number if you wish.

3. Summary of commands.

3.1 Loading Editor/Assembler.

Plug cassette interface cable into cassette external speaker jack. Set volume control to nine. Type P or B for PolyPhase or Byte. Then type ASM. Start tape.

3.2 Editing commands.

EDIT invokes the editor.

3.2.1 Move cursor.

The up-pointing arrow key moves the cursor up to the previous carriage return. (Use CTRL-Q on old keyboards.)

The down arrow moves the cursor down to the next carriage return (CTRL-R).

The right arrow moves the cursor right one character (CTRL-S).

The left arrow moves the cursor left one character (CTRL-T).

3.2.2 Move text.

CTRL-B displays the Beginning of text currently in memory.

CTRL-E displays the End of text.

CTRL-N displays the Next sixteen lines.

CTRL-P displays the Previous sixteen lines.

NOTE: A "line" is the text between two carriage returns.

3.2.3 Create and change text.

Typing on the keyboard creates text. Characters appear in the present cursor location. Delete individual characters with the DELETE or RUB OUT key. —

CTRL-W deletes a Word.

CTRL-X deletes a line.

CTRL-F creates a string to be Found.

ESC searches for a string when used in conjunction with CTRL-F. Then search for the next occurrence by hitting CTRL-C.

Hitting ESC (or ALT MODE) in any circumstances except when finding a string takes you from edit mode to assembly mode.

CTRL-H displays a brief list of edit commands. Hit any key to return to the point you departed from in the text.

3.3 Assembler commands.

~~ASM~~ invokes the assembler.
~~EDIT~~ *editor*

FREE shows the number of currently unused memory locations and the address of the first free byte in memory.

SYMT displays the current assembler symbol table.

LOAD reads tape into memory. Include B or P for Byte or Poly-Phase, plus the name of the desired file between slashes: LOADB /name/ or LOADP /name/.

SAVE records text from memory onto tape. Use B or P and a file name as above.

DUMP records assembled object code from memory onto tape. Use B or P and a name as above, plus From and To addresses: DUMPB /name/ FFFF TTTT or DUMPP /name/ FFFF TTTT . All addresses must be stated in hex. The DUMP command adds an auto-execute record to the dumped program.

CALL begins execution of the program, starting at the address specified: CALL aaaa. A RET (return) instruction in the main program will return the system to assembler mode.

KILL deletes all text from memory (but not Editor/Assembler or symbol table).

HARD selects printer as listing device.

SOFT selects screen only. Disables printer. The assembler defaults to SOFT.

LIST displays the current text on the screen. The display pauses every fifteen lines and advances upon any keystroke.

MNTR invokes the Front Panel Mode.

ASMB assembles the text in memory. Use ASMB plus an origin (where in memory the program will run when used) and an assembly address (where in memory the object code will be put during assembly): ASMB 0000 aaaa. ASMBE lists errors only, ASMBE uses the symbol table previously used, and ASMBES does both. The screen display will pause every fifteen lines, advancing upon any keystroke.

ADDR	cmd	vector
238F	FREE	= 2415
	SYMT	2F75
	LOAD	340A
	SAVE	35E0
	DUMP	3571
	CALL	217E
	ASMB	2B0C
	MNTR	0038
	EDIT -	26D6
23C5	KILL	23DD
	HARD	23E8
	SOFT	23F4
	LIST	23FB

23DD

~3000

ORG
 LON
 ENDS
 LOFF KIL
 EQU SVC
 DB
 DS
 DW
 END

then mnemonics
 APBCDEHLM

MSG5



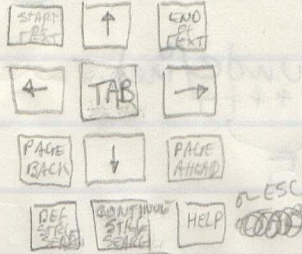
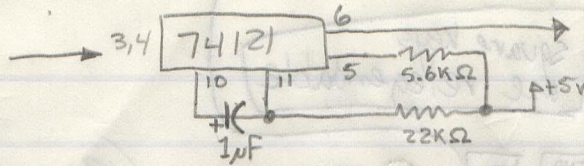
Illegal command
 Bad parameters!
 Bad file name
 augh! checksum error...
 Line too long
 Polymorphic Systems Assembler Version 4.03
 You forgot parameters!
 uh oh... memory full! (you lose...)
 Top of memory is
 Bytes free, starting at
 Too many assembly errors
 Commands ~~mm~~ (Help Commands)
 Symbol table overflow - assembly aborted!

Working...

Load Address? b

Start Address? b

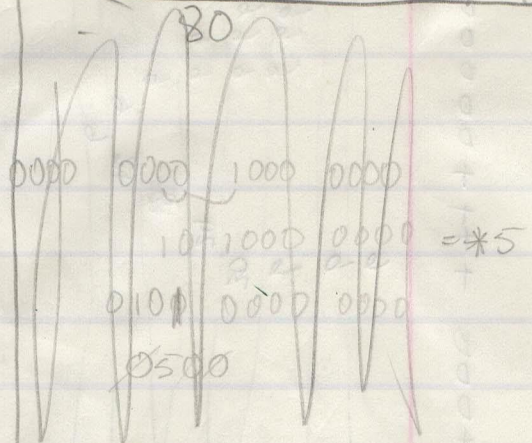
STROBE PULSER (analyse inverts?)



SWTPC
KYBRD

$\uparrow = \text{CTL-Q} = \text{DC1} = 18/X0$
 $\downarrow = \text{CTL-R} = \text{DC2} =$
 $\rightarrow = \text{CTL-S} = \text{DC3} =$
 $\leftarrow = \text{CTL-T} = \text{DC4} =$
 $\text{BEG OF TEXT} = \text{CTL-B} = \text{STX} = 12/X0$
 $\text{END OF TEXT} = \text{CTL-E} = \text{ENQ} = 15/X0$
 $\text{NEXT PAGE} = \text{CTL-N} = \text{SO} =$
 $\text{PREV PAGE} = \text{CTL-P} = \text{DLE} = 10/X1$
 $\text{DEF FIND STRG} = \text{CTL-F} = \text{ACK} = 16/X0$
 $\text{NEXT FIND STRG} = \text{CTL-C} = \text{ETX} = 13/X0$
 $\text{HELP} = \text{CTL-H} = \text{BS} = 15/X3$

Pro K.ybd arranged
for EDIT codes on
BACK of this page



1 PDS + -

HEX IN

PRINT

DUP

SWAP

HEX OUT

DEC OUT

POP
PUSH
PUSH

PUSH H

POP H

MOV E1

MOV D H

POP D

POP H

PUSH D

PUSH H

ADD

POP H

POP D

DA0 D

PUSH H

G03 - top of mem DW at 3BC1
3BC4
3BCE

End of test 3BB1

End of test 3BB3

? 3BCC End of test + 1?

135

ASM 4.6 + other P&J
software

PolyMorphic Assembler System

OPERATING SYSTEM

The PolyMorphic Assembler System is a software package designed to run on the Poly-88 computer. Included in the system is an executive to handle memory and tape files, an Assembler, and a line oriented Editor. To use the system a minimum of 8K of memory should be available.

EXECUTIVE COMMANDS

(CAN key) control X Cancel input line

✓ EXEC HHHH Execute a program

✓ ASMB Assemble a source file to object code

✓ LIST LLLL (opt) List current file

DEL ~~EDIT~~ LLLL LLLL (opt) Delete line or lines of current file

Any four numeric digits enters editor

DISP HHHH HHHH (opt) Display memory

RSEQ Resequence current file

✓ MNTR Go to monitor

✓ LOAD Load source file

✓ SAVE Dump source file

✓ DUMP Dump memory to cassette

✓ FREE Display free memory space

✓ SYMT Displays current symbol table

→ EDIT Enters the character editor - replaces MNTR

To initialize the system, execute it at 2000H. To restart the system without initializing it, execute at 2003H.

The executive has one error message 'WHAT?' indicating an improper command or an error on parameters following the command. The other error that can happen is "CHECKSUM ERROR". This error can occur when a cassette error occurs.

EXEC ####

This command is used to execute a program at address ####.

LIST ####

This command is used to display the lines entered by the user into the file. The output consists of the lines in the file starting at line #### and the next 13 lines. If #### is not specified, then the next 13 lines are displayed. (It is recommended that you enter a zero if you want to display the start of the file.)

DELT #### ##

This command is used to delete lines entered by the user from the file. All lines starting at the first line and continuing up to and including the second number are deleted from the "current" file. If the second number is not specified then only the first is deleted.

LOADP /NAME/**LOADB /NAME/**

Load the file with specified name from cassette. LOADP reads Polyphase tapes, LOADB reads byte format tapes. The names must be enclosed in delimiters (/) as shown, and be 8 characters or less. The syntax must be exactly as shown, with one space separating the B or P and the file delimiting /.

SAVEP /NAME/ LIN1 LIN2**SAVEB /NAME/ LIN1 LIN2**

Save source text inclusive from line number specified as LIN1 to and including the line number specified as LIN2 as the file name given, in byte or Polyphase mode. LIN1 and LIN2 must be given as four digits, with one space between the ending file name delimiter / and the first line number, and one space between the line numbers.

Examples:

>SAVEP /MUNG/ 0010 0450

>SAVEB /CHAIN/ 0100 0300

DUMPP /NAME/ ADR1 ADR2

DUMPB /NAME/ ADR1 ADR2

Dump contents of memory in the address range specified by ADR1 and ADR2 to the named file in byte or Polyphase mode. ADR1 and ADR2 must be specified as four digits and must be separated by one space.

Examples:

>DUMPP /ASM/ 2000 2FFF

>DUMPB /ASM/ 2000 2FFF

SYMT

This command prints out the current SYMBOL TABLE

FREE

This command prints the amount (in hexadecimal) of storage left.

ASMB (E) (S) #### ##

This command is used to assemble a source program located in the file area. The assembler performs the assembly, assigning addresses to the object code starting at the first number. On assembly the object code is placed in memory starting at location number 2. If number 2 is not specified, it is assumed to be the same value as number 1. During pass one errors detected will be displayed, and during pass two a complete listing is produced. If the option "E" is specified in the command, only those lines which contain errors are listed. * If the optional "S" is specified, then the new symbol table will be added onto the previous symbol table. Otherwise it is started at the beginning of the symbol table space.

** USE LOFF INSTEAD † USE ENDS*

NOTE: If "E" and "S" are both specified "E" must be first or else it will be ignored.

TEXT EDITOR

The editor is a line oriented editor which enables the user to easily create program files in the System. Each line is prefaced by a fixed line number which provides for stable line referencing. Since line numbers can range from 0000 to 9999 (decimal)

there are 10,000 lines that can exist in each file. (if enough storage exists). As the user types lines on the keyboard, they are entered into the file area. The editor places all line numbers in sequence and automatically over-writes an existing line in the file if a new line with the same line number is entered by the user.

The editor does not automatically assign line numbers. The user must first, when entering a line of data, enter a line number which will be interpreted as a call to the editor. Valid line numbers must contain four digits; preceding zeros must be included. An entry to the editor is terminated by the carriage return key. No more than 64 characters may be input for one line. All lines are ordered by the ascending numeric sequence of their line numbers. If the user wishes to insert lines after the initial entry is made, it is suggested that the original line numbers be separated by five unit intervals.

ASSEMBLER

When the Assembler is given control by the executive, it proceeds to translate the Symbolic 8080 Assembly Language (source) program into 8080 machine (object) code. Features of the Assembler include:

- Free format source input

- Symbolic addressing, including forward references and relative symbolic references

- Complex expressions may be used as arguments

- Self defining constants

- Multiple constant forms

- Up to 256 eight character symbols

- Reserved names for 8080 registers

- ASCII character code generation

- 8 Pseudo Operations (assembler directives)

The Assembler translates lines in the file into object code. The second character following the line number is considered to be the first source code character position. Hence, the character immediately following the line number should normally be blank. Line numbers are not processed by the Assembler; they are merely reproduced on the listing.

STATEMENTS, COMMENTS, AND PSEUDO OPERATIONS

During pass 1, the Assembler allocates all storage necessary for the translated program and defines the values of all symbols used, by creating a symbol table. The storage allocated for the object code will begin at the first byte dictated by the first parameter in the original Executive ASMB command.

During pass 2, all expressions, symbols, and ASCII constants are evaluated to absolute values and are placed in allocated memory in the appropriate locations. The listing, also produced during pass 2, indicates exactly what data is in each location of memory.

STATEMENTS

Statements may contain either symbolic 8080 machine instructions or pseudo-ops. The structure of such a statement is

NAME	OPERATION	OPERAND	COMMENT
------	-----------	---------	---------

The name-field, if present, must begin in assembler character position one. The symbol in the name-field can contain as many characters as the user wants; however, only the first 8 characters are used in the symbol table to uniquely define a symbol. All symbols in this field must begin with an alphabetic character and may contain no special characters.

The operation-field contains either an 8080 operation mnemonic or an Assembler pseudo-operation code.

The operand-field contains parameters pertaining to the operation-field. If two arguments are present, they must be separated by a comma.

Example:

```
0005 FLOP      MOV    M,B      THIS IS A COMMENT
0015 ;COMMENT LINE
0025           JMP    BEG
0035           CALL   FLOP
0045 BEG       ADI     8+6-4
0055           MOV    A,B
```

All fields are separated and distinguished from one another by the presence of a Tab (control-I) or one or more spaces.

The comment-field is for explanatory remarks. It is reproduced on the listing without processing. See example 0005. Comment lines must start with a semi-colon (;).

SYMBOLIC NAMES

To assign a symbolic name to a statement, one merely places the symbol in the name-field. To leave off the name-field the user skips two or more spaces after the line number and begins the operation field.

If a name is attached to a statement, the assembler assigns it the value of the current location counter. The location counter always holds the address of the next byte to be assembled. The only exception to this is the EQU pseudo-operation. In this case a symbol in the name-field is assigned a value which is contained in the operand-field of the EQU pseudo-operation statement. Example:

```
0057 POTTA     EQU     128
assigns the value of 128 to the name POTTA. This data can then be
used elsewhere in the program as: eg.  ADI     POTTA
```

Names are defined when they appear in the name-field. All defined names may be used as symbolic arguments in the argument-field. See

SYMBOLIC NAMES, cont.

examples on previous page.

In addition to user defined names, the assembler has reserved several symbols, the value of which is predetermined. These names may not be used by the user except in the operand-field. They are (with their value in parentheses):

A the accumulator (7)

B Register B (0)

C Register C (1)

D Register D (2)

E Register E (3)

H Register H (4)

L Register L (5)

M Memory (through H, L) (6)

S Stack pointer (6)

In addition to the above reserved symbols, there is the single special character symbol (\$). This symbol changes in values as the assembly progresses. It is always equated with the value of the program counter after the current instruction is assembled. It may only be used in the operand-field.

Examples:

JMP \$ Means jump to the next location after

MOV A,B instruction; i.e., the MOV instruction.

LDA \$+5 Means load the data at the fifth location

DB 0 after this location. In this case the

DB 1 data has the value of five.

DB 2

DB 3

DB 4

DB 5

RELATIVE SYMBOLIC ADDRESSING

If the name of a particular location is known, a nearby location may be specified using the known name and a numeric offset.

Example:

```

      JMP    BEG
      JPE    BEG+4
      CC     SUB1
      CALL   ADD1
BEG MOV    A,B
      HLT
      MVI    V,'B'
      INR    B

```

In this example the instruction JMP BDG refers to the MOV A,B instruction. The instruction JPE BEG+4 refers to the INR B instruction. BEG+4 means the address BEG plus four bytes. This form of addressing can be used to locate several bytes before or after a named location.

CONSTANTS

The Assembler allows the user to write positive or negative numbers directly in the statement. They will be regarded as decimal constants and their binary equivalents will be used appropriately. All unsigned numbers are considered positive. Decimal constants can be defined using the descriptor "D" after the numeric value. (This is not required, as the default assignment is decimal.)

Hexadecimal constants may be defined using the descriptor "H" after a numeric value. (i.e. 10H, 10H, 3AH, 0F4H).

NOTE: A hexadecimal constant cannot start with the digits A-F.

In this case, a leading 0 must be included. This enables the Assembler to differentiate between a numeric value and a mnemonic symbol.

CONSTANTS, cont.

ASCII constants may be defined by enclosing the ASCII character within single quote marks, i.e., 'C'. For double word constants, two characters may be defined within one quote string.

EXPRESSIONS

An expression is a sequence of one or more symbols, constants or other expressions separated by the arithmetic operators plus or minus. Examples:

PAM+3

ISAB-'A'+5

LOOP+32H-5

Expressions are calculated by using 16 bit arithmetic. All arithmetic is done modulo 65536. Single byte data cannot contain a value greater than 255 or less than -256. Any value outside this range will result in an assembler error.

PSEUDO-OPERATIONS

The pseudo-operations are written as ordinary statements, but they direct the assembler to perform certain functions which do not always develop 8080 machine code. The following section describes the pseudo-ops.

ORG

The ORG statement will set the program counter to the value in the operand-field. The label-field is optional but if present will be equated to the given operand-field.

END

The END statement informs the assembler that the last source statement has been read. The assembler will then start on pass 2, or terminate

END, cont.

the assembly and pass control back to the executive. This pseudo-op is not required as the assembler will stop when an end of file has been reached.

ENDS

The ENDS statement functions exactly like the END statement except it prints the symbol table at the end of pass 2.

LON

The LON will turn the print listing flag on. This results in a full listing from the LON statement until a LOFF statement or the end of assembly.

LOFF

The LOFF statement will turn the print flag off. This results in an "error only" listing until a LON statement or the end of assembly.

EQU

The EQU statement is used to equate a label with an expression.
Example: SAM EQU 12D equates SAM to be equal to 12

DS

The DS statement reserves the number of memory bytes specified by the operation-field.

• SVC

The SVC statement is the same as a RST 7, or Single Step.
Use as a breakpoint for debugging.

KIL

The KIL statement is the same as a RST 0, or Front Panel Reset.
Use as a way of getting out of a jam.

DB

The DB statement generates a single byte constant specified by the operation-field.

DW

The DW statement is used to define two bytes of storage. The evaluated argument will be placed in the two bytes; high order 8 bits in the low order byte and low order 8 bits in the high order byte. This conforms to the Intel format for two byte addresses.

ASSEMBLER ERRORS

The following error flags are output on the assembler listing when the error occurs. Some of the errors are only output during pass one.

- ✓ O Opcode Error
- ✓ L Label Error
- ✓ D Duplicate Label Error
- ✓] Missing Label Error
- ✓ V Value Error
- ✓ U Undefined Symbol
- ✓ S Syntax Error
- ✓ R Register Error
- ✓ A Argument Error

SYMBOL TABLE

The assembler normally allocates a symbol table of 100 (decimal) entries, each entry requiring 10 (decimal) bytes. Decreasing the size of the symbol table frees memory for use by the editor for storing the source text. To increase or change the contents of byte 2006H in the program, and restart the editor/assembler at 2000H. The contents of location 2006H must be greater than 1 and less than 255. This implies a maximum number of symbols at any time of 254.

NORMALLY 64H, or 100 Decimal

1,000 BYTES

SYMBOL TABLE

STARTS AT 3050H

ENDS AT 31FFH when 2006H = 64H

ASM 4.6

2FA0 = DW top of text

2FA2 = DS 4 ASCII Line Number

4 digits

2FA6 = DW ? start addr of last line changed

2F9C = Top of mem

2F9E start of test

TRY P instead of ABCDEHLM

3443

DB

29B0 → memory look up table

KILL

2A00 ~

POLYMORPHIC SYSTEMS SOFTWARE USER GROUP

PolyMorphic Systems would like to encourage the interchange of software between POLY 88 users. In order to do this we will distribute Cassettes of programs we feel are of interest to other users.

Should you develop a program that you would like to see made available to fellow users send us a cassette with a copy of the program in Byte format together with instructions (in Xeroxable form) on how to run the program. Programs we decide to distribute will be included in our price list and will be sold for 10 to 15 dollars. We accept no responsibility for the content or applicability of any program distributed in this manner, and offer this strictly as a service to POLY 88 users.

New **Editing** feature (**replaces MNTR**) for ASM 4.6

IMPORTANT → 'ASM+EDIT' file only has this new feature

To enter the editor from the assembler, type '**EDIT**'. The screen will clear and the first line of the text area will be displayed.

The editor commands are as follows:

ESC - exits back to assembler.

CTL-C - displays the next line.

CTL-F - skips the first character of the current line and displays the remaining characters of the line.

CTL-D - backs up one character of the current line and displays the rest of the line as **CTL-F** does.

CTL-R - displays the previous line if you are looking at a complete line; otherwise it displays the entire current line.

Any other CTL char - is ignored.

All other characters - will be inserted just in front of the current line displayed.

Disassembler

MICROMATICS

8080 DISASSEMBLER

VERSION 01.0

<u>CONTENTS</u>	<u>PAGE</u>
INTRODUCTION	1
DISASSEMBLERS - THEORY OF OPERATION	1
DISASSEMBLER FUNCTIONS	2
◦ Instruction Recognition/Illegal Instructions	2
◦ Pseudo Label Generation	2
◦ Out-Of-Bound Equates	2
◦ Label Table Overflow	2
◦ User Control	2
OPERATING INSTRUCTIONS	3
USER HINTS	4
APPENDIX A - 8080 MNEMONICS	5
APPENDIX B - ASCII CHART	6

MICROMATICS
 P.O. Box 5710
 Columbus, Ohio 43221
 (614) 882-7396

INTRODUCTION

This document is designed to acquaint the user with some of the characteristics and idiosyncrasies of disassemblers and to explain the features and operating procedures of the Micromatics 8080 Disassembler.

While it is not absolutely necessary that the user of this product be an accomplished 8080 assembly language programmer, it is highly recommended that the user possess a good understanding of programming fundamentals and knowledge of the 8080 instruction repertoire.

DISASSEMBLERS - THEORY OF OPERATION

The purpose of any disassembler program is to convert object code instructions to recognizable program instructions/mnemonics.

As we all probably know, the purpose of a computer assembler program is to convert source code instructions (programmer generated statements) into machine executable object code - for example: A source code statement of `MVI A,20H` would be converted to the hexadecimal representation `3E20` by the assembler program. Very simply, the purpose of a disassembler program is to reverse that procedure by taking the hexadecimal representation `3E20` and re-creating the source statement `MVI A,20H`.

Ideally a disassembler program would reverse the assembler process and re-create the source statements for any program 100% accurately and completely. Unfortunately this is not possible because very few programmers use the same programming techniques and since there exists no rigid industry standards for programming methodology. The end result is of course, that computer programs tend to be as individualistic as the persons writing them.

Since it is not practical to program a disassembler to handle all the programming anomalies, they generically tend to be less than error free and generally do not disassemble with complete accuracy any but the simplest of programs. A common disassembler error occurs when a programmer inserts data constants/values between program routines which are subsequently disassembled. Obviously when the disassembler encounters these data constants they are assumed to be instructions and wallah the data is disassembled ! This is really not as much of a problem as it might seem however, especially for a user who is familiar with programming techniques and the 8080 instruction repertoire. Furthermore, most 8080 disassemblers seem to

re-orient themselves very quickly after exiting data and entering object code.

In summary, while disassemblers do not reproduce source code with 100% accuracy they do provide the capability of translating large object code programs to source code statements with a relatively high degree of accuracy and therefore can be of tremendous value to a programmer who wishes to examine a program for which he has no program listing.

DISASSEMBLER FUNCTIONS

- ° Instruction Recognition is the primary purpose of any disassembler. The Micromatics 8080 Disassembler utilizes a 1K table of 8080/8085 mnemonics to translate each object coded instruction into its source code equivalent. Any illegal instructions encountered are displayed as hexadecimal byte constants. The user should note that the 8085 instructions RIM and SIM will be translated whenever hexadecimal values 20 and 30 are encountered.
- ° Pseudo Labels are generated as needed for all instructions which reference memory. As the "memory reference" instructions are encountered a pseudo label is generated and stored into a label table for later resolution. The pseudo labels make the disassembler output more readable and intelligible.
- ° Out-of-Bounds Equates are generated by the Micromatics Disassembler for all those labels whose addresses fall outside of the range of memory being disassembled. This capability should assist the user in identifying other system logic being referenced by the user program.
- ° Label Table Overflow will occur whenever the number of labels encountered exceed the disassembler label table size. This occurrence however, does not abort the disassembly process (see Operating Instructions).
- ° User Control - The user is given the option to continue or terminate the disassembly process at the end of each video screen display.

OPERATING INSTRUCTIONS

- Step 1: Load "DASM" into memory from cassette.
- Step 2: Respond to the query - "Starting address to disassemble:" with any 4 digit hexadecimal address. An invalid response causes the query to be repeated.
- Step 3: Respond to the query - "Ending address to disassemble:" with any 4 digit hexadecimal address. An invalid response causes the query to be repeated. The ending address must be greater than the start address or the message "..... illegal address!" is displayed.
- Step 4: Should the label table overflow, the message ">>> Label table overflow <<<" is displayed followed by an "*" - respond with a carriage return. The disassembly process will continue but all overflowed pseudo labels will be displayed as "LBL???".
- Step 5: When the video screen becomes filled, the message "Continue ?" is displayed. Respond "Y" for yes continue the disassembly process; "N" for no do not continue the disassembly process and go to Step 2.
- Step 6: After the entire disassembly process completes an END statement followed by an "*" is displayed - respond with a carriage return and go to Step 2.

USER HINTS

- Always load the program to be disassembled prior to loading the 8080 Disassembler - DASM.
- The disassembly process will continue even if the label table overflows. However the "overflowed" labels will not have a unique numeric identification.
- The 8085 instructions RIM and SIM are translated for the hexadecimal values 20 and 30.
- The 8080 Disassembler may be restarted at the start address indicated on the cassette label.

8080 MNEMONICS

00 NOP	20 RIM*	40 MOV B,B	60 MOV H,B	80 ADD B	A0 ANA B	C0 RNZ	E0 RPO
01 LXI B,db1e	21 LXI H,db1e	41 MOV B,C	61 MOV H,C	81 ADD C	A1 ANA C	C1 POP B	E1 POP H
02 STAX B	22 SHLD addr	42 MOV B,D	62 MOV H,D	82 ADD D	A2 ANA D	C2 JNZ addr	E2 JPO addr
03 INX B	23 INX H	43 MOV B,E	63 MOV H,E	83 ADD E	A3 ANA E	C3 JMP addr	E3 XTHL
04 INR B	24 INR H	44 MOV B,H	64 MOV H,H	84 ADD H	A4 ANA H	C4 CNZ addr	E4 CPO addr
05 DCR B	25 DCR H	45 MOV B,L	65 MOV H,L	85 ADD L	A5 ANA L	C5 PUSH B	E5 PUSH H
06 MVI B,byte	26 MVI H,byte	46 MOV B,M	66 MOV H,M	86 ADD M	A6 ANA M	C6 ADI byte	E6 ANI byte
07 RLC	27 DAA	47 MOV B,A	67 MOV H,A	87 ADD A	A7 ANA A	C7 RST 0	E7 RST 4
08 ...	28 ...	48 MOV C,B	68 MOV L,B	88 ADC B	A8 XRA B	C8 RZ	E8 RPE
09 DAD B	29 DAD H	49 MOV C,C	69 MOV L,C	89 ADC C	A9 XRA C	C9 RET	E9 PCHL
0A LDAX B	2A LHLD addr	4A MOV C,D	6A MOV L,D	8A ADC D	AA XRA D	CA JZ addr	EA JPE addr
0B DCX B	2B DCX H	4B MOV C,E	6B MOV L,E	8B ADC E	AB XRA E	CB ...	EB XCHG
0C INR C	2C INR L	4C MOV C,H	6C MOV L,H	8C ADC H	AC XRA H	CC CZ addr	EC CPE addr
0D DCR C	2D DCR L	4D MOV C,L	6D MOV L,L	8D ADC L	AD XRA L	CD CALL addr	ED ...
0E MVI C,byte	2E MVI L,byte	4E MOV C,M	6E MOV L,M	8E ADC M	AE XRA M	CE ACI byte	EE XRI byte
0F RRC	2F CMA	4F MOV C,A	6F MOV L,A	8F ADC A	AF XRA A	CF RST 1	EF RST 5
10 ...	30 SIM*	50 MOV D,B	70 MOV M,B	90 SUB B	B0 ORA B	DO RNC addr	FO RP
11 LXI D,db1e	31 LXI SP,db1e	51 MOV D,C	71 MOV M,C	91 SUB C	B1 ORA C	D1 POP D	F1 POP PSW
12 STAX D	32 STA addr	52 MOV D,D	72 MOV M,D	92 SUB D	B2 ORA D	D2 JNC addr	F2 JP addr
13 INX D	33 INX SP	53 MOV D,E	73 MOV M,E	93 SUB E	B3 ORA E	D3 OUT byte	F3 DI
14 INR D	34 INR M	54 MOV D,H	74 MOV M,H	94 SUB H	B4 ORA H	D4 CNC addr	F4 CP addr
15 DCR D	35 DCR M	55 MOV D,L	75 MOV M,L	95 SUB L	B5 ORA L	D5 PUSH D	F5 PUSH PSW
16 MVI D,byte	36 MVI M,byte	56 MOV D,M	76 HLT	96 SUB M	B6 ORA M	D6 SUI byte	F6 ORI byte
17 RAL	37 STC	57 MOV D,A	77 MOV M,A	97 SUB A	B7 ORA A	D7 RST 2	F7 RST 6
18 ...	38 ...	58 MOV E,B	78 MOV A,B	98 SBB B	B8 CMP B	D8 RC	F8 RM
19 DAD D	39 DAD SP	59 MOV E,C	79 MOV A,C	99 SBB C	B9 CMP C	D9 ...	F9 SPHL
1A LDAX D	3A LDA addr	5A MOV E,D	7A MOV A,D	9A SBB D	BA CMP D	DA JC addr	FA JM addr
1B DCX D	3B DCX SP	5B MOV E,E	7B MOV A,E	9B SBB E	BB CMP E	DB IN byte	FB EI
1C INR E	3C INR A	5C MOV E,H	7C MOV A,H	9C SBB H	BC CMP H	DC CC addr	FC CM addr
1D DCR E	3D DCR A	5D MOV E,L	7D MOV A,L	9D SBB L	BD CMP L	DD ...	FD ...
1E MVI E,byte	3E MVI A,byte	5E MOV E,M	7E MOV A,M	9E SBB M	BE CMP M	DE SBI byte	FE CPI byte
1F RAR	3F CMC	5F MOV E,A	7F MOV A,A	9F SBB A	BF CMP A	DF RST 3	FF RST 7

* 8085 Instructions

ASCII CHART

00 NUL	20 SP	40 @	60 \
01 SOH	21 !	41 A	61 a
02 STX	22 "	42 B	62 b
03 ETX	23 #	43 C	63 c
04 EOT	24 \$	44 D	64 d
05 ENQ	25 %	45 E	65 e
06 ACK	26 &	46 F	66 f
07 BEL	27 '	47 G	67 g
08 BS	28 (	48 H	68 h
09 HT	29)	49 I	69 i
0A LF	2A *	4A J	6A j
0B VT	2B +	4B K	6B k
0C FF	2C ,	4C L	6C l
0D CR	2D -	4D M	6D m
0E SO	2E .	4E N	6E n
0F SI	2F /	4F O	6F o
10 DLE	30 0	50 P	70 p
11 DC1	31 1	51 Q	71 q
12 DC2	32 2	52 R	72 r
13 DC3	33 3	53 S	73 s
14 DC4	34 4	54 T	74 t
15 NAK	35 5	55 U	75 u
16 SYN	36 6	56 V	76 v
17 ETB	37 7	57 W	77 w
18 CAN	38 8	58 X	78 x
19 EM	39 9	59 Y	79 y
1A SUB	3A :	5A Z	7A z
1B ESC	3B ;	5B [	7B {
1C FS	3C <	5C \	7C
1D GS	3D =	5D]	7D } (ALT MODE)
1E RS	3E >	5E ^	7E ~
1F US	3F ?	5F _	7F DEL (RUBOUT)

MICROMATICS

P.O. BOX 5710
COLUMBUS, OHIO 43221

November 1978

Dear Customer;

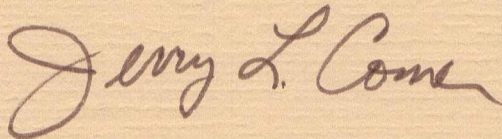
The enclosed Change Notice #01 is provided for the 8080 Disassembler Program (V01.0). The Change Notice describes existing bug(s) and provides either source or object patches to remedy same.

We will issue change notices as required to correct any known "software bugs" which come to our attention.

Micromatics is dedicated to providing quality software products to our customers.

Thanks for your patronage.

Cordially,

A handwritten signature in dark ink, reading "Jerry L. Comer". The signature is written in a cursive style with a large, stylized "J" and "C".

Jerry L. Comer
Manager

JLC/jh

8080 DISASSEMBLER (V01.0) OBJECT CODE PATCHES

SYMPTOM: Pseudo Labels generated for memory reference instructions may not be uniquely identified at disassembly time.
(i.e. LBL??? may be displayed instead of LBL001)

SOLUTION: Modify SRCH subroutine to test label # field in tag table for end of table instead of label address field.

The following address byte locations flagged as "nn" and "pp" are determined by the system configuration of your Disassembler.

System 1 (Start address = 3100H); Substitute 35H for "nn" and 37H for "pp"

System 2 (Start address = 5100H); Substitute 55H for "nn" and 57H for "pp"

System 3 (Start address = 4D00H); Substitute 51H for "nn" and 53H for "pp"

System 4 (Start address = 7100H); Substitute 75H for "nn" and 77H for "pp"

System 5 (Start address = 6D00H); Substitute 71H for "nn" and 73H for "pp"

System 6 (Start address = 6500H); Substitute 69H for "nn" and 6BH for "pp"

P A T C H E S

<u>ADDRESS</u>	<u>CONTENTS</u>	<u>CHANGE TO</u>
nn9C	7E	23
nn9D	FE	23
nn9E	FF	7E
nn9F	CABEnn	C377pp
pp77	unused	FEFF
pp79	unused	C8
pp7A	unused	2B
pp7B	unused	2B
pp7C	unused	7E
pp7D	unused	BB
pp7E	unused	CAAAnn
pp81	unused	C3A6nn

8080 Programs

June 7, 1976

Dear PCC Dragon,

Just about a month ago, I was using a Fluke 1900A Frequency counter to check the accuracy of the tempered musical scale that *ALPHA-NUMERIC MUSIC generates on my Altair computer. The frequency for Alpha-Numeric music is determined by a binary number in a look-up table in memory from address 003,024 to 003,147. The frequencies for octaves one through three was found slightly high and the reason was that the binary numbers were low by one count in most cases due to a mathematical error. To correct this problem here is a new series of binary numbers to replace the frequency table from 003,024 to 003,075.

Address	Octal bytes
003,024	373,355,340,323,310,274,262,250,236,225,215,205,176,167
003,042	176,167,160,152,144,136,131,124,117,113,107,103,077,073
003,060	077,073,070,065,062,057,055,052,050,046,043,041,040,036

In musical notation, each octave starts with the number for B below C and ends with C for the next higher octave for a total of 14 numbers. For those who would like to calculate their own numbers for the first three octaves, here is the equations:

N=Binary number that will be decrement in Reg.-C

F=Frequency

μ =Micro (10^{-6})

$$N = ((1/2F) + 12\mu s)/64.5\mu s$$

$$F = \frac{1}{2(64.5\mu s(N) - 12\mu s)}$$

The equation in the Alpha-Numeric Music article for "N" is for octaves four through six.

Yours truly,
Malcolm Wright

* Alpha-Numeric Music with Amplitude Control, software by Malcolm Wright being distributed by PCC.



\$2

ALPHA-NUMERIC MUSIC

with

AMPLITUDE CONTROL

copyright 1975 by Malcolm T. Wright

Includes 1976 Supplements
(April 13, May 17, & June 7)

PEOPLE'S COMPUTER COMPANY
BOX 310, MENLO PARK CA 94025

Nov. 7, 1975

Alpha-Numeric Music with Amplitude Control.

It's interesting that I should make a music routine my first major programming effort, but maybe I was hoping to make the computer more of a crowd pleaser with the limited peripheral hardware I had at the time. It was obvious the programming problems would be challenging, since I am just learning the machine instructions for the 8080 CPU and I cannot read music. Of course, one can approach programming and music with the same basic premise: "they are both just a coding & decoding problem"! I am sure that many of you will be able to simplify and improve on this music program after you understand the basic principles that lie within it.

The Alpha-Numeric Music program that will be discussed in the following pages was written for the Altair 8800 computer with a ASCII (upper & lower case) keyboard and one 8-bit output port feeding an amplifier thru some resistors. The output port circuit will be described later on in this article.

After writing nine different music routines for my 8080 CPU, it became obvious that the frequency range and sound quality of the music would be determined by writing a flexible square-wave output routine. The square-wave routine should be made short to increase running speed which will determine the maximum frequency, and it should be written with CPU timing in mind to eliminate noises due to long routines that would upset the symmetry of the square-wave. Therefore the greatest evolution in all of the different music programs was a section called Play-Note which generated the binary coded square-wave.

Play-Note

The routine called Play-Note uses all of the registers in the 8080 CPU and also the memory stack. Play-Note uses register-C to set the time for a half-cycle of the square wave to be generated. Registers D&E are used to count the number of half-cycles which will set the duration of the note played. Register-B is used to count the number of segments that make-up an envelope of a note. Registers H&L are used to point at a table in memory which gives the values for the amplitude of the envelope of the square-wave. Register-A is loaded with the amplitude of the envelope. The stack is used to save the present working address of the music table and save intermediate register values for the Play-Note routine.

CTL-J = CTL-Q

Sample MusicThe Sound of Silenceby Paul Simon

CTRL-T 2, CTRL-V 2, CTRL-E 5, CTRL-R 4, 4, D, D, D, F, F, A, A, /
W, G, /, O, R, C, C, C, E, E, G, G, /, W, F, /, O, R, F, F, F, A, A, S, C, C, /
H, D, C, /, Q, R, 4, O, F, F, A, A, S, C, C, /, H, D, C, /, 4, Q, R, O, F, F, /
S, O, D, Q, CTRL-A, D, Q, D, O, D, E, /, O, F, Q, CTRL-A, F, O, E, Q, CTRL-A, D, /
CTRL-Q, H, CTRL-A, C, O, D, C, /, W, 4, A, /, O, R, F, F, F, /
S, H, CTRL-A, C, 4, O, R, E, /, F, Q, CTRL-A, D, H, D, /, CTRL-E, 3
CTRL-S, 5, H, CTRL-A, C, 4, O, D, C, /, 3, H, A, Q, CTRL-A, D, 4, O, E, /
O, F, Q, CTRL-A, F, F, S, F, F, /, S, H, CTRL-A, C, O, R, 4, E, /
F, O, CTRL-A, D, H, D, /, CTRL-E, 4, Q, R, O, D, D, F, F, A, A, /, W, G, /
Q, R, O, C, C, E, E, G, G, /, W, F, /, CTRL-R, 1, Q, R, O, F, F, A, A, S, C, C, /
Q, CTRL-A, D, O, C, H, C, /, 4, CTRL-S, Q, R, O, F, F, Q, A, S, O, C, C, /
Q, D, Q, D, D, D, Q, E, /, O, F, F, F, F, E, Q, CTRL-A, D, /
H, C, O, C, C, S, D, D, O, C, /, 4, H, CTRL-A, A, O, R, E, /, F, Q, CTRL-A, F, Q, O, F, F, /
S, W, CTRL-A, C, O, R, 4, E, /, F, Q, CTRL-A, D, H, D, /, D, /, L

April 13, 1976

May 17, 1976

To All Music Buffs,

The interest that has been shown in my Alpha Numeric Music routine is surprising. I have been approached by more music enthusiasts wanting to encode a favorite tune of theirs on my computer. I am starting to collect quite a few popular pieces for my computer's library through the musical help of some more talented friends. It looks like there might be enough interest in my music routine that people may soon want to exchange songs, so let me make sure we have a standard here.

(1) *On page 20 under NOTES, item 6, I want to make the exclamation mark the standard for a flat. This will allow people with teletypes to exchange music with other ASCII keyboard systems.

(2) I am sorry that I didn't know that a Control-J is the same as a LINE FEED, so let's change it to a CTRL-Q. Therefore at address 000,162 (page 10) change byte 012 to 021 (octal).

(3) Final item. I have found a way to improve the transfer of the desired envelope within my program with a few changes.

Routine Address	Change only the following
001,343	072,121,002 MOD1: LDA SaveV
001,347	176 MOV A,M
001,350	117 MOV C,A
001,363	171 MOV A,C
001,366	000 NOP
001,370	372,376,001 JM Cont4
001,373	303,343,001 JMP MOD1

These changes will improve loading and coding of the ton envelopes. Instead of using "even parity" to mark the end of each envelope, you will now use only the "MSB". This changes the envelope table to:

003,161	007,007,007,007,007,007,007,207
003,172	007,007,007,007,000,000,000,200
003,203	007,007,000,007,007,000,007,200
003,214	007,006,005,004,003,002,001,200
003,225	001,002,003,004,005,006,007,200
003,236	001,002,005,007,000,005,002,001,200

Good Luck,

Malcolm T. Wright

Pages from an article called "Alpha Numeric Music with Amplitude Control" by Malcolm Wright, being distributed by PCC.

Dear PCC Dragon,

I am sorry to say that my "Alpha-Numeric Music" article, which you are distributing, has two typing errors in the machine instruction listing. A friend pointed out that at address 001,101 it should be:

303,110,001 (good)
and not 303,110,000 (bad)
and I found that at address 000,203 which is:
376,163 (small-s)
should be 376,123 (capital-s)

For those who have my music routine up and running here is another piece of music for your enjoyment.
Prelude in C by J.S. Bach, encoded by Walter White.

CTRL-T, 2, CTRL-E, 3, CTRL-V, 2
CTRL-R, 1, 4, S, C, E, G, S, C, E, 4, G, S, C, E, CTRL-S, /
CTRL-R, 1, 4, C, D, A, S, D, F, A, A, S, D, F, CTRL-S, /
CTRL-R, 1, 3, 9, 4, D, G, S, D, F, 4, G, S, D, F, CTRL-S, /
CTRL-R, 1, 4, C, E, C, S, C, E, 4, G, S, C, E, CTRL-S, /
CTRL-V, 3, CTRL-R, 1, 4, C, E, A, S, E, A, 4, A, S, E, A, CTRL-S, /
CTRL-V, 2, CTRL-R, 1, 4, C, D, F, #, A, S, D, 4, F, #, A, S, D, CTRL-S, /
CTRL-V, 3, CTRL-R, 1, 3, 8, 4, D, G, S, D, C, 4, G, S, D, G, CTRL-S, /
CTRL-V, 2, CTRL-R, 1, 3, 8, 4, C, E, G, S, C, 4, E, G, S, C, CTRL-S, /
CTRL-V, 1, CTRL-R, 1, 3, A, 4, C, E, G, S, C, 4, E, G, S, C, CTRL-S, /
CTRL-R, 1, 3, D, A, 4, D, F, #, 5, C, 4, D, F, #, 5, C, CTRL-S, /
CTRL-R, 1, 3, G, 8, 4, D, C, B, D, C, B, CTRL-S, /
CTRL-R, 1, 3, G, B, 1, 4, E, G, S, C, #, 4, E, G, S, C, #, CTRL-V, 2, CTRL-S, /
CTRL-R, 1, 3, F, A, 4, D, A, S, D, 4, D, A, S, D, CTRL-V, 3, CTRL-S, /
CTRL-R, 1, 3, F, A, 1, 4, D, E, S, D, F, B, CTRL-V, 2, CTRL-S, /
CTRL-R, 1, 3, E, G, 4, C, G, S, C, 4, C, G, S, C, CTRL-S, /
CTRL-R, 1, 3, E, F, A, 4, C, F, 3, A, 4, C, F, CTRL-S, /
CTRL-R, 1, 3, D, F, A, 4, C, F, 3, A, 4, C, F, CTRL-S, /
CTRL-V, 1, CTRL-R, 1, 2, G, 3, D, G, B, 4, F, 3, G, B, 4, F, CTRL-S, /
CTRL-R, 1, 3, C, E, G, 4, C, E, 3, G, 4, C, E, CTRL-V, 2, CTRL-S, /
CTRL-R, 1, 3, C, G, B, 1, 4, C, E, 3, B, 1, 4, C, E, CTRL-V, 3, CTRL-S, /
CTRL-R, 1, 2, F, 3, F, A, 4, C, E, 3, A, 4, C, E, CTRL-V, 2, CTRL-S, /
CTRL-R, 1, 2, F, #, 3, C, A, 4, C, E, 1, 3, A, 4, C, E, 1, CTRL-V, 1, CTRL-S, /
CTRL-R, 1, 2, G, 3, E, 1, B, 4, C, E, 1, 3, B, 4, C, E, 1, CTRL-S, /
CTRL-R, 1, 2, A, 1, 3, F, S, 4, C, E, 1, 3, B, 4, C, E, 1, CTRL-S, /
CTRL-V, 2, CTRL-R, 1, 2, G, 3, F, G, B, 4, D, 3, G, B, 4, D, CTRL-S, /
CTRL-V, 3, CTRL-R, 1, 2, G, 3, E, G, 4, C, E, 3, G, 4, C, E, CTRL-S, /
CTRL-R, 1, 2, G, 3, D, G, 4, C, F, 3, G, 4, C, F, CTRL-S, /
CTRL-R, 1, 2, G, 3, D, G, B, 4, F, 3, G, B, 4, F, CTRL-S, /
CTRL-R, 1, 2, G, 3, E, 1, A, 4, C, F, #, 3, A, 4, C, F, #, CTRL-S, /
CTRL-R, 1, 2, G, 3, E, G, 4, C, G, 3, G, 4, C, G, CTRL-S, /
CTRL-R, 1, 2, G, 3, D, G, B, 4, F, 3, G, B, 4, F, CTRL-S, /
CTRL-V, 1, CTRL-R, 1, 2, C, 3, C, B, 1, 4, E, 3, G, B, 1, 4, E, CTRL-S, /
CTRL-T, 1, 2, C, 3, C, F, A, 4, C, F, 3, A, 4, C, F, A, F, D, F, D, /
CTRL-V, 1, 2, C, B, CTRL-V, 2, 4, G, B, S, D, F, D, 4, B, S, D, 4, G, B, /
CTRL-V, 1, D, F, E, D, CTRL-V, 2, /, 4, W, C, S, R, /, L

Yours truly,

Malcolm T. Wright

Nov. 7 1975

Music Decoding Sheet

Octave 6

Octave 5

Octave 4

Octave 3

Octave 2

Octave 1

Note... C D E F G A B C D E F G A B C D E F G A B

o ...Whole note(W)

d ...Half note(H)

♩ ...Quarter note(Q)

♪ ...Eighth note(O)

♫ ...Sixteenth note(S)

♭ ...Thirtysecond note(T) ,

♩ ...Increase note duration by 50%. (CTRL-A)

⌋ ...Repeat (between these brackets repeat the measures as indicated.) (CTRL-R, CTRL-J, CTRL-S)

⏏ ...Quarter rest(QR)

♩ ...Eighth rest(OR)

— ...Half rest(HR)

— ...Whole rest(WR)

b ...Flat (b)

...Sharp (#)

♮ ...Natural (cancels a sharp or flat)

⌋ ...Measure separator(/)

The heart of Play-Note can be reduced to the following:

```

Cycle:  MVI  C,Freq.1  ; half-cycle duration no.
        OUT  1
Loop:   DCR  C          ; half-cycle timer
        JNZ  Loop
        XRA  M          ; squarewave generator
        JMP  Cycle
  
```

This small routine is a basic software timer. The duration of this routine is predictable since each machine instruction has an exact execution time set by the 2MHz crystal within the computer. By placing the right binary number after MVI,C, and adding on the execution time for each instruction, any one of 256 different frequencies can be generated at output 1.

The duration of the square-wave produced by the above routine has no limits, so to be able to control the duration of the square-wave two more jump instructions were added.

```

Cycle:  LXI  D,Duration ; load note duration no.
        MVI  C,Freq.1
        OUT  1
Loop:   DCR  C
        JNZ  Loop
        XRA  M
        DCR  E          ; lower byte of duration
        JNZ  Cycle
        DCR  D          ; upper byte of duration
        JNZ  Cycle
        RET
  
```

Registers D&E form a double-precision counter of the square wave cycles and their preset value (LXI,D) will determine the length of the note.

Now to add some variety to the sound quality of the notes that this routine will generate, the ability to change the amplitude of the square-wave will be added to Play-Note.

```

Level:  MVI  B,Seg.      ; load no. of segments
        MOV  A,M         ; get segment value
        LXI  D,Duration
Cycle:  MVI  C,Freq.1
        OUT  1
Loop:   DCR  C
        JNZ  Loop
        XRA  M
        DCR  E
        JNZ  Cycle
        DCR  D
        JNZ  Cycle
        INX  H           ; get next segment
        DCR  B
        JNZ  Level      ; finished all segments?
        RET
  
```


The variable amplitude (envelope) is stored in a table in memory and registers H&L are used as a pointer to scan through the table.

This routine has one problem, it can only be used for an even number of half-cycles. An odd number of cycles set by LXI,D will cause popping sounds during an envelope due to two positive half-cycles adding together giving a cycle twice as long between each segment. Therefore let's add a routine to correct for odd cycles.

```

Level:  MVI    B,Seg.
        MOV    A,M
Dura:   LXI    D,Duration
Cycle:  MVI    C,Freq.1
        OUT    1
Loop:   DCR    C
        JNZ    Loop
        XRA    M
        DCR    E
        JNZ    Cycle
        DCR    D
        JNZ    Cycle
        INX    H
        DCR    B
        JZ     Exit
        CPI    Zero      ;odd cycle checker
        MVI    A,Zero
        JZ     Dura
        JMP    Level
Exit:   RET

```

Now the above routine has to be corrected for timing. The 8080 CPU within my computer is controlled by a memory circuit to provide two 0.5 microsecond wait cycles for each memory byte read or write sequence. The Play-Note routine and the main music program was written for two wait cycles, but once the basic principles are understood it can be modified. The final Play-Note routine has some time wasters*(do nothing routines) in it and an extra output instruction to correct the timing.

```

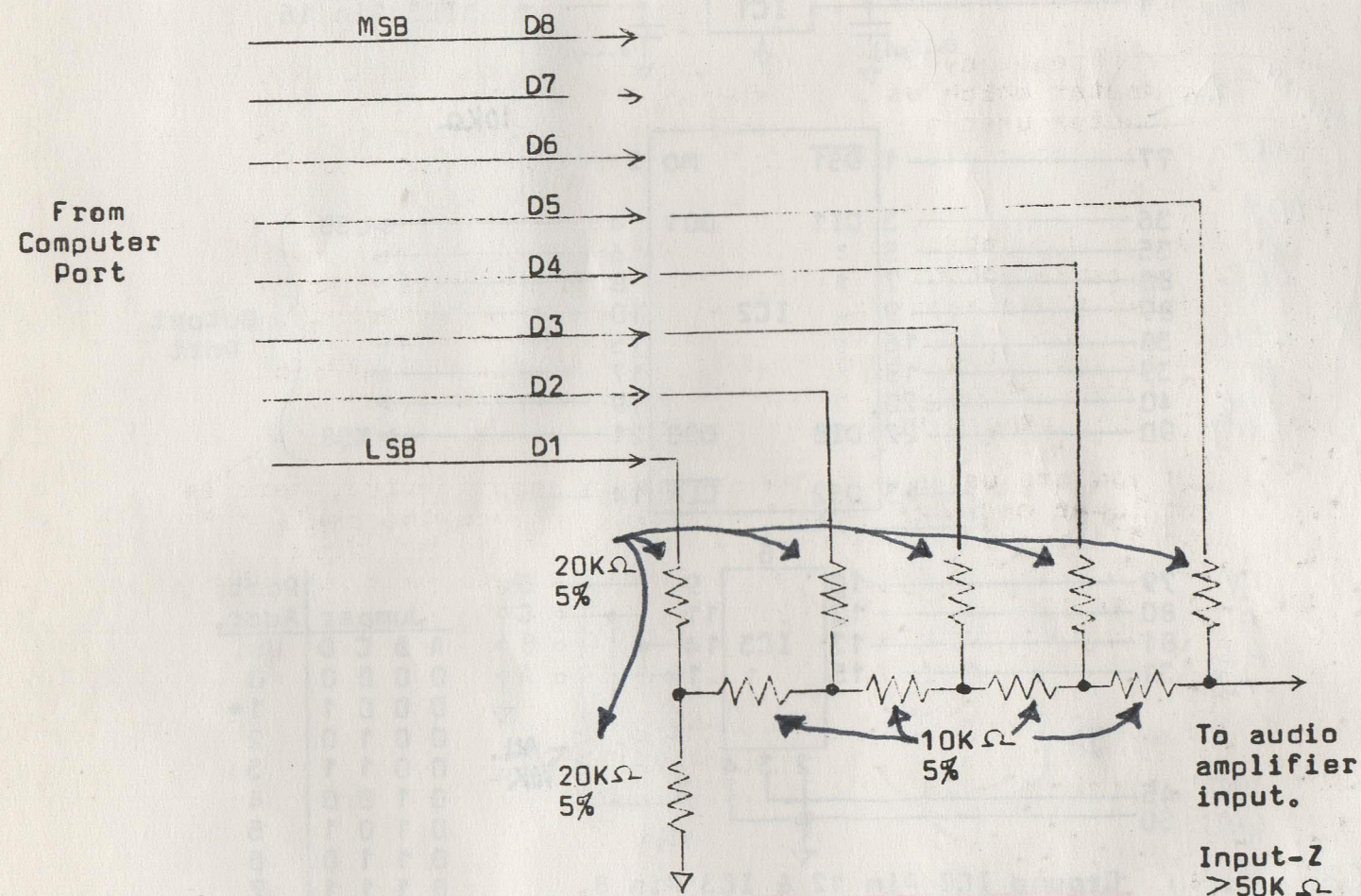
* Play-Note: LXI    H,Env.table ;load envelope table
              MVI    B,Seg.
              MOV    A,M
              LXI    D,Duration
Loop2:       MOV    A,A *
              JMP    Cycle *
Cycle:       MVI    C,Freq.1
              OUT    1
Loop1:       DCR    C
              JNZ    Loop1
              XRA    M
              DCR    E
              JNZ    Loop2

```

Notes

- (1) The Alpha-numeric music program was written as one very large subroutine, so to run the program you must CALL it from another program.
- (2) The address of the stack pointer is not set(LXI,SP) by the music program, but the stack is used so be sure that it is not at a location within the program or the music table when you run. A maximum of ten bytes are placed in the stack.
- (3) The "Frequency table" was written for the Altair 8800 computer which uses two memory wait cycles. If your computer uses a different number of wait cycles, the freq. table will have to be changed to match the new timing.
- (4) The music table starts at H=003 and L=314. To load the table you will have to use some supporting software(like a monitor) or load it by hand (Groan!).
- (5) When using the repeat codes (CTRL-R,CTRL-J & CTRL-S) you can not nest(put repeats inside of repeats) a musical repeat with this program.
- (6) If you are using a Teletype for an input device,there is no lower case "b" for flat notes. Change the small "b" to "!" by modifying address 000,377 & 001,000 from 376,142 to 376,041. The exclamation mark will now flat a note when used in the music table listing.

Digital-Analog-Converter needed for output port 1



```

DCR D
JNZ Cycle
ORI Zero *
OUT 1
INX H
DCR B
JZ Exit
LXI D,Duration
MVI C,Freq.2 **
CPI Zero
MVI A,Zero
JZ Loop1
DCR C *
DCR C *
NOP *
MOV A,M
JMP Loop1
RET

```

Exit:

*.....time waster or corrector

**.....Loaded by music program with a number which will correct the timing.

Timing

The half-cycle timer, which is the section of the routine labeled "Loop1", uses two instructions with a total time of 11.5 microseconds. Everytime DCR,C is reduced to zero, the duration registers D or E are decremented and the routine returns to Loop1 with an addition offset time of 41 microseconds. The basic timing equation then would be:

N=Binary number that will be decrement in Reg-C.

F=Frequency

H=Half-cycle period of frequency

$$N(11.5\mu s) + 41\mu s = H$$

$$2(H) = 1/F$$

$$N = \left[\frac{1}{2F} - 41\mu s \right] / 11.5\mu s$$

To generate the frequency of the tempered scale for middle C (261.6HZ), the value for "N" would be 162.64. The processor requires "N" to be an integer, therefore the resultant frequency for N=163 is about 261.03HZ. The frequency error is due to the fact that the cycle can not be resolved any finer than 11.5μs.

The duration of each of the envelope's segments is set by registers D&E. The approximate time needed to fetch each segment is 88.5 microseconds. The equation for duration:

D=Double precision binary number in registers D&E.

S=Duration of one segment.

E=Number of envelope segments.

$$H = \frac{1}{2F}$$

$$S = D(H) + 88.5\mu s$$

$$\text{Note duration} = E(S)$$

In trying to generate a frequency range of six octaves the Play-Note routine would give an upper frequency of 2093 HZ with reasonable accuracy, but would not go below 168.2 HZ. Therefore the music program was given the ability to change the jump address after DCR,C of Loop1 to link up with another time-waster routine extending the Loop1 time to 64.5 microseconds.

Time extender

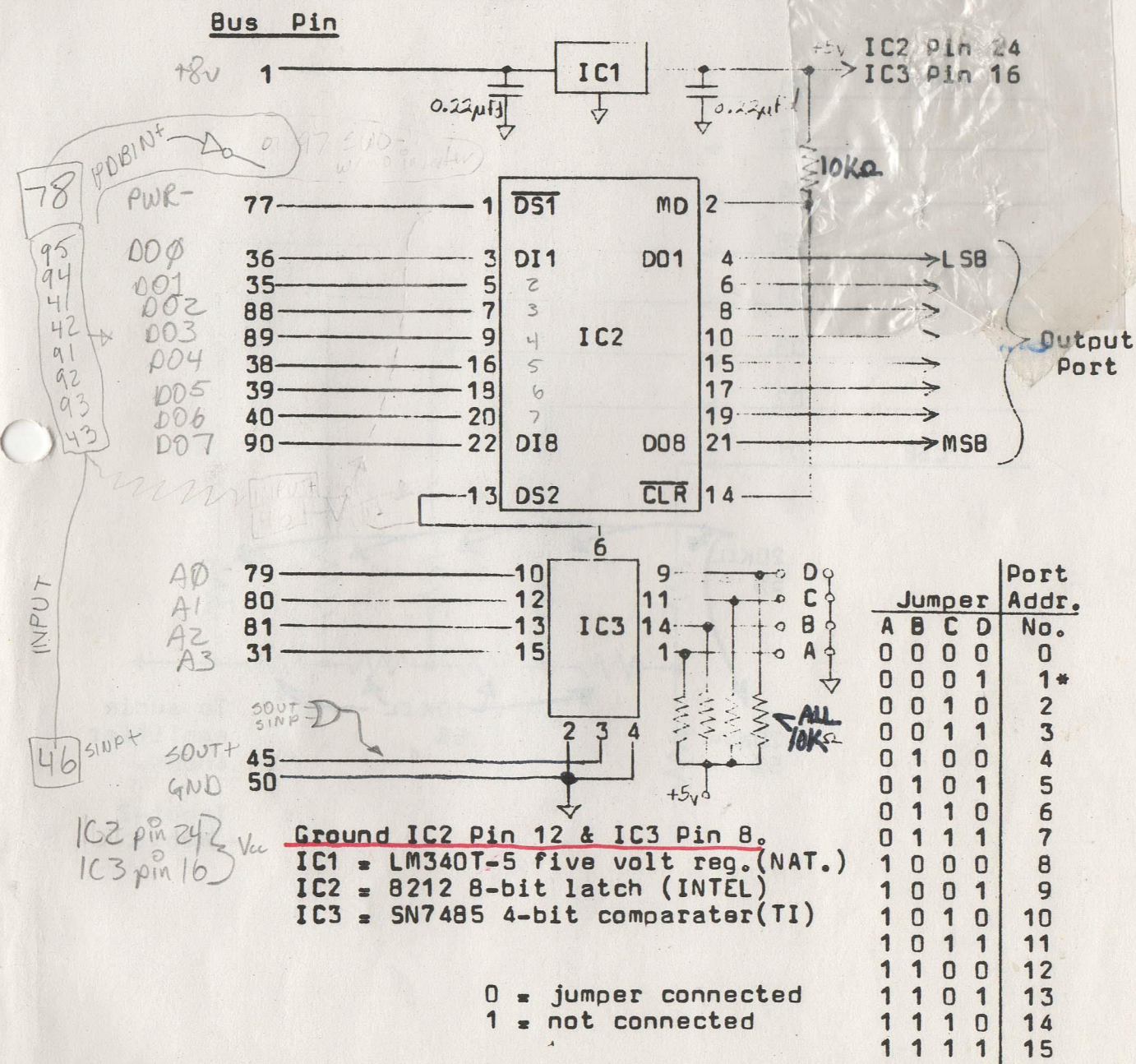
```

Count:  PUSH PSW      ;save Reg-A & zero flag
        MVI  A,002
        DCR  A
        JNZ  Count
        POP  PSW      ;restore Reg-A & flags.
        JMP  Loop1

```

Hardware

To hear the results of this software music program, a port with a 5-bit DAC(Digital-to-Analog Converter) will have to be added to the computer. If you do not have a 8-bit parallel output port, one can be built with three IC's.



*....I used port 1, but this can be changed to any port 0 thru 15 very easily if you already have a peripheral device at 1. Note: Be sure to change all output instructions, and the LXI,H after CALL Play-Note, to agree with your port choice.

see next page also

FOR REVISED ENVELOPE TABLE, SEE PG. 22

Address	"Envelope table"	CTRL-E	
003,161	007,007,007,007,007,007,007,007,000	0	
172	007,007,007,007,007,200,200,200,000	1	
203	007,007,200,007,007,200,007,007,000	2	
214	007,206,205,004,203,002,001,200,000	3	
225	001,002,203,004,205,206,007,200,000	4	
236	001,002,205,007,007,205,002,001,000	5	
247	000,000,000,000,000,000,000,000,000	6	spare
260	000,000,000,000,000,000,000,000,000	7	"
271	000,000,000,000,000,000,000,000,000	8	"
302	000,000,000,000,000,000,000,000,000	9	"
313	000		

"Music table"
003,314Start of User written music.

For spare envelopes, try these

#6 002,004,007,007,007,005,003,001,200
#7 003,007,007,007,006,005,003,001,200

Music Table Coding

Now in the beginning the music programs were difficult to code into the computer because all of the frequencies had to be looked up and decoded from tables by the User into binary values. The Alpha-Numeric Music program removes some of the difficult decoding tasks by placing tables within the memory and letting the CPU do the looking up. The coding of this high level language is as follows:

Notes

Type.....C,D,E,F,G,A,B
Modifiers(suffix): b....flat note
#....sharp note

Durations

Type W=whole note O=eighth note
H=half note S=sixteenth note
Q=quarter note T=thirtysecond note
Modifier(suffix): CTRL-A....increase the duration of note by 50%

Tempo

Type CTRL-T
Modifiers(suffix): 1....100 Beats/Min.
2....125 Beats/Min.
3....150 Beats/Min.
4....Spare, programmed by User.

Volume

Type CTRL-V
Modifiers(suffix): 1....Low
2....Medium
3....High

Special Repeat Codes

- [CTRL-R... This code followed by a repeat-no. should be placed in front of the coded measures the User wishes to repeat.
-] CTRL-Q... This code will decrement the repeat-no. and jump the music to one location beyond the CTRL-S code. if the repeat-no. equals zero.
- : CTRL-S... This code must follow the coded measures that are being repeated. CTRL-S will decrement the repeat number and if it is not zero, jump the music back to one location beyond the CTRL-R code.

Octaves

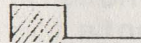
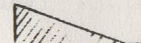
Type 1...C(32.71HZ) up to B.
2...C(65.41HZ) " " "
3...C(130.8HZ) " " "
4...C(261.6HZ) " " "
5...C(523.3HZ) " " "
6...C(1046.5HZ) " " "

Other Codes

Type R.....a rest (no sound)
 S.....If typed when the music program
 is running it will stop and
 RETURN to the User's program
 that CALLED Alpha-Numeric Music.
 L.....Must be the last character in the
 music table.

Envelopes

Type CTRL-E
 Modifiers(suffix):

0.....  constant level
 1.....  staccato
 2.....  triplet
 3.....  bell sound
 4.....  ?
 5.....  like tremolo
 6..... Spare
 7... "
 8... "
 9... "

Some of the encoding characters for the music table
 were patterned after an article written by Dr. Knowlton
 and his work in pasadena in which he interfaced a pipe
 organ with a computer for a couple of concerts.

Address	Mach. Instr.	Label	Mnemonic	Comments
02,173	366,000		ORI 000 ; (TE)	
175	323,001	Out2:	OUT 001	
177	043		INX H	
002,200	005		DCR B	
201	312,227,002		JZ Exit	
204	021,157,001	Dur2:	LXI D	
207	016,237		MVI C,Freq2 ; (TE)	
211	376,000		CPI 000 ; even-odd cycle tester	
213	076,000		MVI A,000	
215	312,156,002		JZ Loop1	
220	015	ChgT:	DCR C ; (TE)	
221	015		DCR C ; (TE)	
222	000		NOP ; (TE)	
223	176		MOV A,M	
224	303,156,002		JMP Loop1	
227	311	Exit:	RET	
230	000		NOP	
231	365		PUSH PSW	
232	076,002		MVI A,002	
234	075	Count:	DCR A	
235	302,234,002		JNZ Count	
240	361		POP PSW	
241	303,156,002		JMP Loop1	
"Tempo table"				
002,244	017,002,056,002,117,002,163,002,230,002,277,002,351,002			100 beats
262	025,003,104,003,166,003,253,003,342,003,035,004,134,004			per min.
300	245,001,277,001,331,001,365,001,023,002,063,002,124,002			125 beats
316	167,002,235,002,305,002,357,002,034,003,113,003,175,003			per min.
334	137,001,164,001,212,001,242,001,272,001,325,001,361,001			150 beats
352	016,002,056,002,117,002,162,002,227,002,276,002,350,002			per min.
370	000,000,000,000,000,000,000,000,000,000,000,000,000			spare
003,006	000,000,000,000,000,000,000,000,000,000,000,000,000			tempo
"Freq.(note) table"				
003,024	372,354,337,323,307,273,261,247,236,225,214,204,175,166			1st Oct.
042	175,166,157,151,143,135,130,123,116,112,106,102,076,073			2nd Oct.
060	076,073,067,064,061,056,054,051,047,045,043,041,037,035			3rd Oct.
076	255,243,231,220,210,200,171,162,153,145,137,132,124,120			4th Oct.
114	124,120,113,106,102,076,073,067,064,061,056,053,050,046			5th Oct.
132	050,046,044,041,037,035,034,032,030,027,025,024,022,021			6th Oct.
"Special table"				
003,150	007,007,007,007,007,007,007,000			

FOR 1ST THREE OCTAVES SEE REVISED TABLE AT PG 23 (BACK PG.)

Address	Mach. Instr.	Label	Mnemonic	Comments
002,031	062,127,002		STA SaveR	
034	042,130,002		SHLD Start	
037	311		RET Next	
040	072,127,002	Jout:	LDA SaveR ; Jump-out routine	
043	075		DCR A	
044	062,127,002		STA SaveR ; repeat no. minus one	
047	376,000		CPI 000	
051	302,057,002		JNZ Cont5	
054	052,132,002		LHLD End	
057	311	Cont 5:	RET Next	
060	042,132,002	Stop:	SHLD End ; Stop-repeat routine	
063	072,127,002		LDA SaveR	
066	075		DCR A	
067	062,127,002		STA SaveR ; repeat no. minus one	
072	376,000		CPI 000	
074	312,102,002		JZ Cont6	
077	052,130,002		LHLD Start	
102	311	Cont6:	RET Next	
002,103	345	Rest:	PUSH H ; Rest routine	
104	041,000,000		LXI H ; clear H&L	
107	042,154,002		SHLD Out1	
112	042,175,002		SHLD Out2	
115	303,116,001		JMP Opmet	
"Save table"				
120	002	SaveT		; tempo no.
121	003	SaveV		; volume no.
122	001	SaveE		; envelope no.
123	004	SaveO		; octave no.
124	003	SaveD		; duration no.
125	001	SaveN		; half-tone no.
126	000	SaveF		; CTRL-A flag.
127	001	SaveR		; repeat no.
130	326,003	Start		; store H&L
132	373,003	End		; store H&L
134	000	NOP		
"Play-Note"				
135	041,150,003		LXI H ; start of special table	
140	006,011		MVI B,011 ; no. of segments	
142	176		MOV A,M	
143	021,157,001	Dur1:	LXI D ; duration of note.	
146	177	Loop2:	MOV A,A ; time equalizer (TE)	
147	303,152,002		JMP Cycle (TE)	
152	016,243	Cycle:	MVI C,Freq1	
154	323,001	Out1:	OUT 001	
156	015	Loop1	DCR C	
157	302,156,002		JNZ Loop1	
162	256		XRA M	
163	035		DCR E	
164	302,146,002		JNZ Loop2	
167	025		DCR D	
170	302,152,002		JNZ Cycle	

Main Program Architecture

The music can be broken down into four main parts (1) character search, 144 bytes, (2) perform character routine, 385 bytes, (3) table look-up, 307 bytes, and (4) play-note routine, 71 bytes.

The character search is done two ways:

Jump method

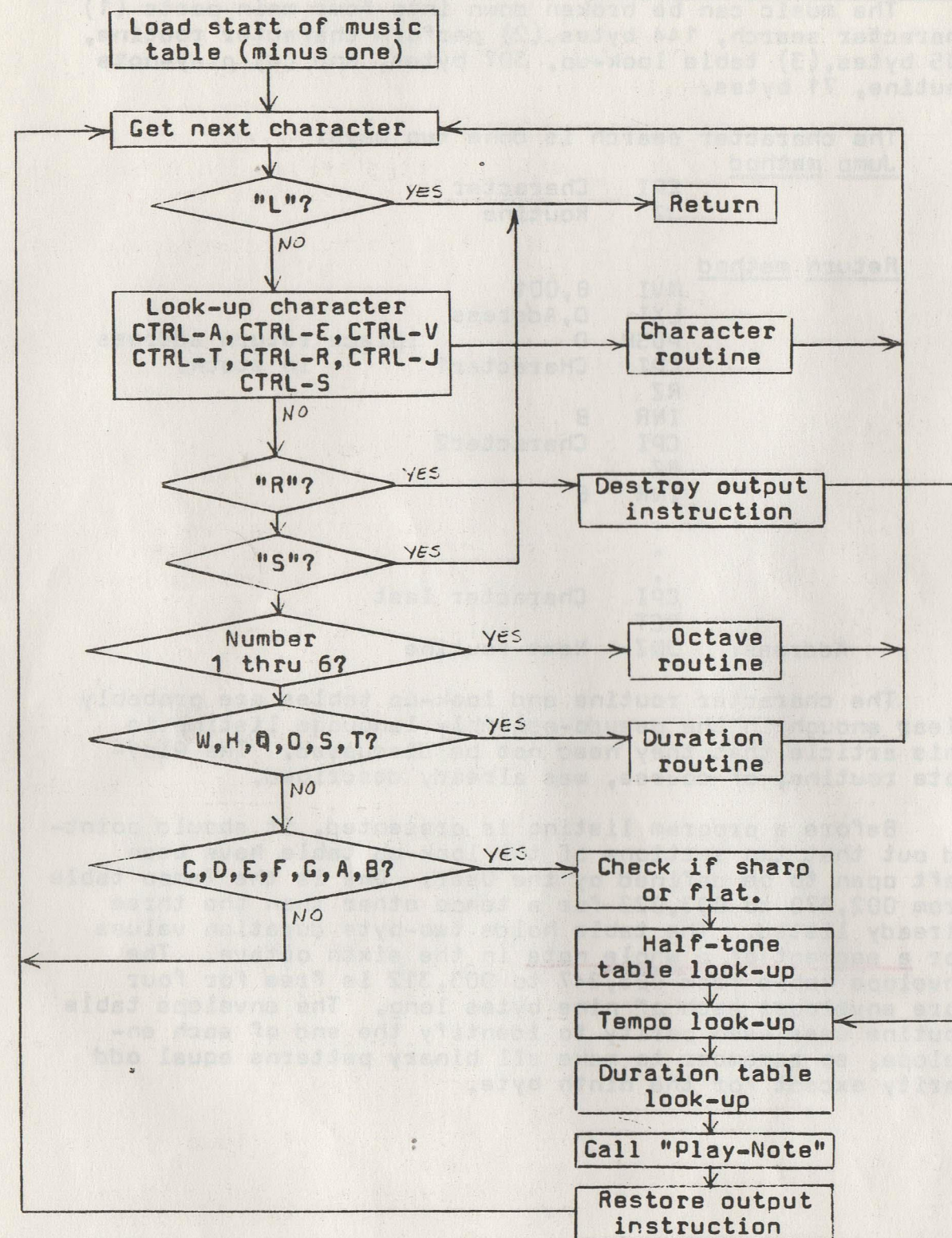
CPI	Character
JZ	Routine

Return method

MVI	B,001	
LXI	D,Address	
PUSH	D	; place return address in stack.
CPI	Character1	
RZ		
INR	B	
CPI	Character2	
RZ		
INR	B	
.		
.		
.		
CPI	Character last	
RET		
Address:	JNZ	Next routine

The character routine and look-up tables are probably clear enough in the pseudo-assembly language listing in this article that they need not be discussed. The Play-Note routine, of course, was already described.

Before a program listing is presented, it should be pointed out that two sections of the look-up table have been left open to be defined by the User. One is the Tempo table from 002,370 to 003,023 for a tempo other than the three already listed. The table holds two-byte duration values for a segment of a whole note in the sixth octave. The Envelope table from 003,247 to 003,312 is free for four more envelopes each of nine bytes long. The envelope table routine uses even parity to identify the end of each envelope, so remember to make all binary patterns equal odd parity except for the ninth byte.



BASIC FLOW CHART

Address	Mach. Instr.	Label	Mnemonic	Comments
001,301	043	Envel:	INX H	;Envelope routine
302	176		MOV A,M	
303	346,017		ANI 017	
305	074		INR A	
306	062,122,002		STA SaveE	
311	315,315,001		CALL Spenv;construct special envelope	
314	311		RET Next	
315	345	Spenv:	PUSH H	
316	021,011,000		LXI D	;load length of envelope
321	041,161,003		LXI H	;1st address of the envel. table
324	072,122,002		LDA SaveE	
327	107		MOV B,A	
330	005	Bak5:	DCR B	
331	312,340,001		JZ Cont2	
334	031		DAD D	
335	303,330,001		JMP Bak5	
340	021,150,003	Cont2:	LXI D	;1st address of special table
343	072,121,002	Mod1:	LDA SaveV	
346	107		MOV B,A	
347	110 176		MOV C,B A,M	
350	176 117		MOV A,M	;get envelope value
351	005	Bak6:	DCR B	
352	312,362,001		JZ Cont3	
355	267		ORA A	;clear carry.
356	027		RAL	
357	303,351,001		JMP Bak6	
362	022	Cont3:	STAX D	
363	101 171		MOV AB,C	
364	023		INX D	
365	043		INX H	
366	176 000		MOV A,M	NOP
367	267		ORA A	;set parity flag.
370	372 352,376,001		JPE JM Cont4	
373	303,351,001		JMP Bak6	MOD1
376	341 343	Cont4:	POP H	
377	311		RET	
002,000	043	Vol:	INX H	;Volume routine
001	176		MOV A,M	
002	346,007		ANI 007	
004	062,121,002		STA SaveV	
007	315,315,001		CALL Spenv	
012	311		RET Next	
013	043	Tem:	INX H	;Tempo routine
014	176		MOV A,M	
015	346,007		ANI 007	
017	062,120,002		STA SaveT	
022	311		RET Next	
023	000		NOP	
024	043	Rep:	INX H	;Repeat routine
025	176		MOV A,M	
026	346,017		ANI 017	
030	074		INR A	

Address	Mach. Instr.	Label	Mnemonic	Comments
001,140	072,125,002	GetD:	LDA	SaveN ;note duration search.
143	007		RLC	;multiply by two reg.-A
144	137		MOV	E,A
145	031		DAD	D
146	136		MOV	E,M ;get lower half of duration.
147	043		INX	H
150	126		MOV	D,M ;get upper half of duration.
151	072,123,002		LDA	SaveD
154	057		CMA	;convert 1 thru 6 to 6 thru 1.
155	346,007		ANI	007
157	107		MOV	B,A
160	005	Bak3:	DCR	B
161	312,202,001		JZ	Cont1
164	315,172,001		CALL	DPSR ;double-precision shift-right.
167	303,160,001		JMP	Bak3
172	257	DPSR:	XRA	A ;reset carry. Start of DPSR.
173	172		MOV	A,D
174	037		RAR	
175	127		MOV	D,A
176	173		MOV	A,E
177	037		RAR	
200	137		MOV	E,A
201	311		RET	
001,202	072,124,002	Cont1:	LDA	SaveD
205	107		MOV	B,A
206	005	Bak4:	DCR	B
207	312,220,001		JZ	GetF
212	315,172,001		CALL	DPSR
215	303,206,001		JMP	Bak4
220	072,126,002	GetF:	LDA	SaveF
223	376,001		CPI	001 ;is it a CTRL-A?
225	041,000,000		LXI	H ;clear H & L.
230	302,237,001		JNZ	NoFg
233	031		DAD	D ;set H & L equal to D & E.
234	315,172,001		CALL	DPSR
237	031	NoFg:	DAD	D
240	353		XCHG	;set D & E equal to H & L.
241	257		XRA	A ;set reg.-A to zero.
242	272		CMP	D ;don't let reg-D and
243	302,253,001		JNZ	Inrd
246	273		CMP	E ← reg-E equal zero!
247	302,253,001		JNZ	Inrd
252	034		INR	E
253	024	Inrd:	INR	D
254	353		XCHG	;put D&E into H&L.
255	042,144,002		SHLD	Dur1+1
260	042,205,002		SHLD	Dur2+1
263	315,135,002		CALL	Play-Note
266	041,323,001		LXI	H ;load H&L with a output instr.
271	042,154,002		SHLD	Out1
274	042,175,002		SHLD	Out2
277	341		POP	H ;restore H&L to music table.
300	311		RET	Next

Alpha-Numeric Music with Amplitude Control

Address	Mach. Instr.	Label	Mnemonic	Comments
000,100	041,313,003		LXI	H ;starting address of music table
103	043	Next:	INX	H
104	176		MOV	A,M
105	346,177		ANI	177 ;erase parity
107	021,103,000		LXI	D ;load stack with return address
112	325		PUSH	D for character search.
113	000		NOP	
114	376,114		CPI	114 ;is the character an "L"?
116	302,124,000		JNZ	CTRLA
121	063	Leave:	INX	SP
122	063		INX	SP
123	311		RET	;return to "mother" program or resident monitor.
124	376,001	CTRLA:	CPI	001 ; is it a CTRL-A?
126	302,135,000		JNZ	CTRLE
131	062,126,002		STA	;save CTRL-A as flag.
134	311		RET	;return to "Next".
135	376,005	CTRLE:	CPI	005 ; is it a CTRL-E?
137	312,301,001		JZ	Envel ;go to envelope routine.
142	376,026		CPI	026 ; is it a CTRL-V?
144	312,000,002		JZ	Vol ;go to volume routine.
147	376,024		CPI	024 ; is it a CTRL-T?
151	312,013,002		JZ	Tem ;go to tempo routine.
154	376,022		CPI	022 ; is it a CTRL-R?
156	312,024,002		JZ	Rep ;go to repeat routine.
161	376,012,021		CPI	012 ; is it a CTRL-J?
163	312,040,002		JZ	Jout;go to jump-out routine.
166	376,023		CPI	023 ; is it a CTRL-S?
170	312,060,002		JZ	Stop;go to stop-repeat routine.
173	376,122		CPI	122 ; is it a "R"?
175	312,103,002		JZ	Rest;go to rest routine.
000,200	107		MOV	B,A
201	333,001		INP	001
203	376,163,123		CPI	123 ; is it a "S"?
205	170		MOV	A,B
206	312,121,000		JZ	Leave ; leave music program.
211	363		DI	
212	376,061		CPI	061 ;is it a 1?,Octave no. search.
214	332,255,000		JC	Dura
217	376,067		CPI	067 ;is it a 7?
221	322,255,000		JNC	Dura
224	346,007		ANI	007 ;save only lower 3 bits.
226	062,123,002		STA	SaveD
231	345		PUSH	H ;save music table address.
232	376,004		CPI	004 ; is it the 4th octave.
234	332,245,000		JC	Chglp
237	041,156,002		LXI	H,Fast
242	303,250,000		JMP	Instrld
245	041,231,002	Chglp:	LXI	H,Slow
250	042,160,002	Instrld:	SHLD	;load in new jump address
253	341		POP	H
254	311		RET	;return to "Next".

11

Address	Mach. Instr.	Label	Mnemonic	Comments
000,255	006,001	Dura:	MVI	B,001 ;duration character search.
257	021,312,000		LXI	D,Addr1
262	325		PUSH	D
263	376,127		CPI	127 ;is it a "W"?
265	310		RZ	
266	004		INR	B
267	376,110		CPI	110 ;is it a "H"?
271	310		RZ	
272	004		INR	B
273	376,121		CPI	121 ;is it a "Q"?
275	310		RZ	
276	004		INR	B
277	376,117		CPI	117 ;is it a "O"?
301	310		RZ	
302	004		INR	B
303	376,123		CPI	123 ;is it a "S"?
305	310		RZ	
306	004		INR	B
307	376,124		CPI	124 ;is it a "T"?
311	311		RET	Addr1
000,312	302,326,000	Addr1:	JNZ	Note
315	170		MOV	A,B
316	062,124,002		STA	SaveD
321	257		XRA	A ;set register-A to zero.
322	062,126,002		STA	SaveF ;erase CTRL-A flag.
325	311		RET	;return to "Next".
326	006,001	Note:	MVI	B,001 ;note character search.
330	021,374,000		LXI	D,Addr2
333	325		PUSH	D
334	376,103		CPI	103 ;is it a "C"?
336	310		RZ	
337	004		INR	B
340	004		INR	B
341	376,104		CPI	104 ;is it a "D"?
343	310		RZ	
344	004		INR	B
345	004		INR	B
346	376,105		CPI	105 ;is it a "E"?
350	310		RZ	
351	004		INR	B
352	376,106		CPI	106 ;is it a "F"?
354	310		RZ	
355	004		INR	B
356	004		INR	B
357	376,107		CPI	107 ;is it a "G"?
361	310		RZ	
362	004		INR	B
363	004		INR	B
364	376,101		CPI	101 ;is it a "A"?
366	310		RZ	
367	004		INR	B
370	004		INR	B

12

Address	Mach. Instr.	Label	Mnemonic	Comments
000,371	376,102		CPI	102 ;is it a "B"?
373	311		RET	Addr2
374	300	Addr2:	RNZ	Next
375	043		INX	H
376	176		MOV	A,M
377	376,142		CPI	142 ;is it a "b"?
001,001	302,010,001		JNZ	Snote
004	005		DCR	B
005	303,022,001		JMP	Stonote
010	376,043	Snote:	CPI	043 ; is it a "#"?
012	302,021,001		JNZ	Notfd
015	004		INR	B
016	303,022,001		JMP	Stonote
021	053	Notfd:	DCX	H
022	170	Stonote:	MOV	A,B
023	062,125,002		STA	SaveN
026	345		PUSH	H
027	072,123,002		LDA	Save0 ;start freq. look-up.
032	107		MOV	B,A
033	041,024,003		LXI	H ;1st address of freq. table
036	021,016,000		LXI	D ;No. of bytes in a octave.
041	005	Bak1:	DCR	B
042	312,051,001		JZ	GetN
045	031		DAD	D
046	303,041,001		JMP	Bak1
051	072,125,002	GetN:	LDA	SaveN
054	137		MOV	E,A
055	031		DAD	D
056	176		MOV	A,M
057	062,153,002		STA	Freq1;save freq. value in Play-Note.
062	107		MOV	B,A
063	072,123,002		LDA	Save0
066	376,004		CPI	004
070	170		MOV	A,B
071	332,104,001		JC	Low ;jump if it is octave 1,2 or 3.
074	326,004		SUI	004 ;subtract 4 from freq. value.
076	041,015,015		LXI	H ;change Play-Note instruction.
001,101	303,110,000 001		JMP	Store
104	075	Low:	DCR	A
105	041,000,000		LXI	H ;change Play-Note instr.
110	062,210,002	Store:	STA	Freq2;save freq.value in Play-Note.
113	042,220,002		SHLD	ChgT
116	072,120,002	Opmet:	LDA	SaveT ;start tempo look-up.
121	107		MOV	B,A
122	041,244,002		LXI	H ;1st address of tempo table
125	021,034,000		LXI	D ;No. of bytes in a octave.
130	005	Bak2:	DCR	B
131	312,140,001		JZ	GetD
134	031		DAD	D
135	303,130,001		JMP	Bak2

TSC

8080

SOFTWARE

PACKAGE

PD80-1

TSC

TECHNICAL SYSTEMS CONSULTANTS
BOX 2574 W. LAFAYETTE INDIANA 47906

TSC

TSC

Technical Systems Consultants
Box 2574 W. Lafayette IN 47906

TSC

8080

SOFTWARE PACKAGE 1

Copyright 1977 (c) by

Technical Systems Consultants
Box 2574 W. Lafayette IN 47906

all rights reserved

IMPORTANT INFORMATION

The programs in this package were designed for maximum flexibility so they could be run on many 8080 based machines. The programs begin on page 1016 of memory but can be relocated by merely changing all addresses to reflect the page numbers you are using. The stack is initialized immediately upon entry (stack address at address 1001₁₆) by LXI SP. You should substitute your stack address before attempting to run any program.

These programs were assembled using the I/O routine address of the TSC 8080 monitor. You should substitute the address of your I/O routines before starting the program. This change is made at the beginning of each program at the JMP linkage. The routine requirements are as follows:

- | | | |
|---------|----|--|
| INCH | -- | Input one ASCII coded character to the A register. <u>The parity bit must be zero.</u> Flags may be modified but no other registers may be altered. |
| OUTCH | - | Output the ASCII coded character contained in the A register. <u>The parity bit is zero.</u> Flags may be modified <u>but no other registers may be altered.</u> |
| MONITOR | - | This is the entry point to your monitor and used as an exit from the program. If you have no monitor this JMP may be replaced with a HLT. |

When using the Random Number Generator or a program containing the same you must be sure that the two bytes used for storage (RNDM, RNDM+1) are not both zero. If so, set one or both to some non-zero value.


```

*
* TSC RANDOM NUMBER GENERATOR FOR 8080
*
*
* COPYRIGHT (C) 1977 BY
* TECHNICAL SYSTEM CONSULTANTS
* BOX 2574 W. LAFAYETTE, IN. 47906
*
* THE TSC RANDOM NUMBER GENERATOR SUPPLIES THE
* RANDOMNESS REQUIRED BY MANY PROGRAMS. THE
* PROGRAM IS BASED ON A PSEUDO-RANDOM SHIFT
* REGISTER APPROACH WHICH HAS BEEN WIDELY PUB-
* LISHED. THE PROGRAM USES TWO BYTES (RNDM AND
* RNDM+1) FOR THE SHIFT REGISTER. THE ONLY
* REQUIREMENT IS THAT AT LEAST ONE OF THE BYTES
* MUST BE NON-ZERO, OTHERWISE IT WILL NOT RUN.
* A CALL TO THIS ROUTINE (CALL RANDM) RETURNS A
* RANDOM NUMBER IN THE ACCUMULATOR WITH THE
* FLAGS MODIFIED BUT NO OTHER REGISTERS AFFECTED.
* THE USER MUST HAVE SETUP A STACK PRIOR TO
* CALLING THIS ROUTINE.
*
*

```

1000		ORG	1000H	
1000		RNDM DS	2	
1002	E5	RANDM PUSH	H	SAVE H AND L
1003	C5	PUSH	B	SAVE B REGISTER
1004	21 00 10	LXI	H,RNDM	POINT TO RNDM
1007	06 08	MVI	B,8	SET LOOP COUNTER
1009	7E	RLOOP MOV	A,M	GET MS BYTE OF RNDM
100A	07	RLC		ALIGN BITS 13 AND 14
100B	AE	XRA	M	XOR WITH RNDM
100C	17	RAL		GET BIT 13.XOR.14
100D	17	RAL		INTO CARRY BIT
100E	23	INX	H	
100F	7E	MOV	A,M	ROTATE BOTH
1010	17	RAL		BYTES OF RNDM
1011	77	MOV	M,A	PUTTING CARRY
1012	2B	DCX	H	INTO LSB
1013	7E	MOV	A,M	
1014	17	RAL		
1015	77	MOV	M,A	
1016	05	DCR	B	
1017	C2 09 10	JNZ	RLOOP	DO LOOP 8 TIMES
101A	C1	POP	B	RESTORE B REGISTER
101B	E1	POP	H	RESTORE H AND L
101C	C9	RET		
101D		END		

SYMBOLIC REFERENCE TABLE.

RANDM	1002	RNDM	1000
RLOOP	1009		

:10100200E5C521001006087E07AE1717237E177765

:0B1012002B7E177705C20910C1E1C951

**

**

:0000000000

*
* HANGMAN FOR 8080
*
*
* COPYRIGHT (C) 1977 BY
* TECHNICAL SYSTEMS CONSULTANTS
* BOX 2574 W. LAFAYETTE, IN. 47906
*
* THIS IS AN 8080 MICROCOMPUTER IMPLE-
* MENTATION OF THE WELL KNOWN GAME HANGMAN.
* IN THIS GAME THE COMPUTER SELECTS A WORD
* AT RANDOM FROM THE LIST SUPPLIED AND
* PROMPTS YOU FOR A GUESS. THE OBJECT
* IS TO GUESS, ONE LETTER AT A TIME, WHAT
* THE WORD IS BEFORE THE COMPUTER SPELLS
* OUT THE WORD HANGMAN. IT WILL ADD A
* SUCCEEDING LETTER TO HANGMAN FOR
* EACH OF YOUR INCORRECT GUESSES, SO
* YOU CAN ONLY MISS 7 TIMES. WHEN YOU
* GUESS A LETTER CORRECTLY THE COMPUTER WILL
* INDICATE THE POSITION OF THE FIRST SUCH
* LETTER IN THE WORD.
*
* SEE ATTACHED SAMPLE OUTPUT FOR AN EXAMPLE.
*
* THE WORD LIST USED STARTS AT WDLST (ADDRESS
* 11F7 HEX). THE WORD LIST MUST BEGIN WITH
* THE HEX CODE 04 AND EACH WORD IS FOLLOWED
* BY A 04 WITH A MAXIMUM OF 9 LETTERS IN ANY
* WORD. THE WORD LIST CAN BE CHANGED BY
* INSERTING THE ASCII FOR EACH LETTER IN
* SUCCESSIVE MEMORY LOCATIONS AND FOLLOWING
* EACH WORD WITH A 04. IF THE NUMBER OF
* WORDS IS CHANGED, THE HEXIDEcimal EQUIV-
* ALENT OF THE NEW WORD COUNT MUST BE STORED
* IN LOCATION 1008 HEX.
*
* AFTER ANY WORD IS SELECTED FROM THE LIST
* IT'S LETTERS ARE ZEROED OUT TO PRECLUDE
* THE POSSIBILITY OF MULTIPLE OCCURRENCES OF
* ANY WORD IN ANY ONE SESSION. BECAUSE OF
* THIS, WHEN ALL WORDS HAVE BEEN USED THE
* COMPUTER WILL NOT PROCEED WITH THE GAME.
* TO PLAY AGAIN, RELOAD THE PROGRAM (OR
* JUST THE WORD LIST). IF YOU WISH TO
* SUPPRESS THE DESTROYING OF THE WORDS,
* CHANGE MEMORY LOCATION 10AF HEX TO A NOP
* (00 HEX), AND LOCATION 1066 HEX TO A JMP
* (C3 HEX).
*
* THIS PROGRAM ASSUMES THAT THE USER HAS AN
* I/O TERMINAL OF SOME TYPE AND ROUTINES TO
* PRINT AN ASCII CHARACTER FROM THE A REGISTER

* AND INPUT AN ASCII CHARACTER INTO THE A
 * REGISTER. THE USER MUST LOAD THE ADDRESS
 * OF THE OUTPUT ROUTINE AT LOCATION 1017 HEX
 * AND THE ADDRESS OF THE INPUT ROUTINE AT
 * 101A HEX (IN NORMAL 8080 FORM). ALSO, THE
 * USER MUST INSTALL THE ENTRY ADDRESS OF HIS
 * MONITOR AT 101D HEX. THE SOURCE LISTING
 * IS SUPPLIED WITH THE ADDRESSES SET FOR THE
 * TSC 8080 MONITOR. THE STACK ADDRESS IS
 * SPECIFIED AT LOCATION 1001 HEX. IF IT IS
 * NECESSARY THIS ADDRESS CAN BE MODIFIED TO
 * SUIT YOUR PARTICULAR SYSTEM.

*
 * THE STARTING ADDRESS OF THIS PROGRAM IS
 * 1000 HEX. WHEN THE COMPUTER ASKS TO PLAY
 * AGAIN, TYPE Y FOR YES OR N FOR NO.
 * CAUTION MUST BE EXERCISED THAT BYTES RNDM
 * AND RNDM+1 (LOCATIONS 1006 AND 1007 HEX)
 * ARE NOT BOTH ZERO. IF THIS IS THE CASE,
 * CHANGE ONE OR BOTH BYTES TO SOME NON-ZERO
 * VALUE BEFORE RUNNING THE PROGRAM.

*
 *

1000	00 00 00	ORG	1000H	
1000	31 FF 1F	LXI	S,1FFFFH	SETUP STACK
1003	C3 55 10	JMP	BEGIN	START THE GAME
1006		RNDM DS	2	
1008	20	WDCNT DB	20H	
1009		HGCNT DS	1	
100A		WDBUF DS	10	
1014	3E 20	OUTS MVI	A,20H	PRINT A SPACE

* JUMP TO EXTERNAL ROUTINES

1016	C3 F9 81	OUTCH JMP	081F9H	
1019	C3 05 02	INCH JMP	08205H	C3 DD 24
101C	C3 1E 00	MONTR JMP	0801EH	

* PRINT ROUTINES

101F	CD 16 20	PCHR CALL	OUTCH	PRINT CHARACTER
1022	23	PDNXT INX	H	
1023	7E	PDATA MOV	A,M	GET CHARACTER
1024	FE 04	CPI	4	CHECK FOR EOT
1026	C2 1F 20	JNZ	PCHR	
1029	C9	RET		
102A	23	PSNXT INX	H	POINT TO NEXT STRING
102B	CD 31 20	PSTRG CALL	PCRLF	PRINT CR AND LF
102E	C3 23 20	JMP	PDATA	PRINT STRING
1031	E5	PCRLF PUSH	H	SAVE H AND L
1032	21 72 21	LXI	H,CRLF	
1035	CD 23 20	CALL	PDATA	PRINT CR AND LF
1038	E1	POP	H	RESTORE H AND L
1039	C9	RET		

* RANDOM NUMBER GENERATOR

103A	E5	RANDM	PUSH	H	SAVE H AND L
103B	C5		PUSH	B	SAVE B REGISTER
103C	21 06 20		LXI	H,RNDM	POINT TO RNDM
103F	06 08		MVI	B,8	SET LOOP COUNTER
1041	7E	RLOOP	MOV	A,M	GET MS BYTE OF RNDM
1042	07		RLC		ALIGN BITS 13 AND 14
1043	AE		XRA	M	XOR WITH RNDM
1044	17		RAL		GET BIT 13.XOR.14
1045	17		RAL		INTO CARRY BIT
1046	23		INX	H	
1047	7E		MOV	A,M	ROTATE BOTH
1048	17		RAL		BYTES OF RNDM
1049	77		MOV	M,A	PUTTING CARRY
104A	2B		DCX	H	INTO LSB
104B	7E		MOV	A,M	
104C	17		RAL		
104D	77		MOV	M,A	
104E	05		DCR	B	
104F	C2 41 20		JNZ	RLOOP	DO LOOP 8 TIMES
1052	C1		POP	B	RESTORE B REGISTER
1053	E1		POP	H	RESTORE H AND L
1054	C9		RET		
1055	21 08 20	BEGIN	LXI	H,WDCNT	GET INITIAL WORD
1058	4E		MOV	C,M	COUNT INTO C
1059	CD 31 20		CALL	PCRLF	
105C	21 79 21		LXI	H,HANG	PRINT 'HANGMAN'
105F	CD 2B 20		CALL	PSTRG	
1062	CD 31 20		CALL	PCRLF	
1065	0D	START	DCR	C	DEC. WORD COUNT
1066	F2 72 20		JMP → JP	HEADR	ANY WORDS LEFT?
1069	21 DD 21	<i>if don't want word list destroyed</i>	LXI	H,END	IF NOT, REPORT IT
106C	CD 2B 20		CALL	PSTRG	
106F	C3 1C 20		JMP	MONTR	AND EXIT
1072	21 8A 21	HEADR	LXI	H,IAM	PRINT HEADER
1075	CD 2B 20		CALL	PSTRG	
1078	CD 2A 20		CALL	PSNXT	
107B	CD 3A 20	GETRN	CALL	RANDM	GET RANDOM NUMBER
107E	E6 3F		ANI	3FH	
1080	CA 7B 20		JZ	GETRN	ZERO NOT VALID
1083	47		MOV	B,A	
1084	3A 08 20		LDA	WDCNT	
1087	B8		CMP	B	COMPARE WORD COUNT
1088	FA 7B 20		JM	GETRN	IF NOT IN RANGE, GETRN
108B	11 0A 20		LXI	D,WDBUF	POINT D/E TO WORD BUFFER
108E	21 F6 21		LXI	H,WDLST-1	POINT H/L TO WORD LIST
1091	23	PTNXT	INX	H	
1092	7E		MOV	A,M	GET A CHARACTER
1093	FE 04		CPI	04	IS IT END CHAR
1095	C2 91 20		JNZ	PTNXT	IF NOT POINT TO NEXT
1098	05		DCR	B	ONE WORD PASSED
1099	C2 91 20		JNZ	PTNXT	
109C	AF		XRA	A	WE'RE AT SELECTED WORD

109D	47		MOV	B,A	ZERO LETTER COUNTER
109E	32 09 20		STA	HGCNT	ZERO THE ERROR COUNT
10A1	23	GTNXT	INX	H	ALIGN POINTER
10A2	7E		MOV	A,M	GET A CHARACTER
10A3	B7		ORA	A	IF ZERO, WORD WAS USED
10A4	CA 7B 20		JZ	GETRN	SO GET ANOTHER WORD
10A7	12		STAX	D	TRANSFER TO WORD BUFFER
10A8	FE 04		CPI	04	WAS IT END CHARACTER
10AA	CA B4 20		JZ	DONE1	IF SO, ALL DONE
10AD	04		INR	B	KICK LETTER COUNTER
10AE	13		INX	D	POINT NEXT
10AF	36 00		MVI	M,0	ZERO LETTER IN WDLST
10B1	C3 A1 20		JMP	GTNXT	GET ANOTHER
10B4	3E 30	DONE1	MVI	A,30H	
10B6	80		ADD	B	MAKE LETTER COUNT ASCII
10B7	CD 16 20		CALL	OUTCH	OUTPUT LETTER COUNT
10BA	21 AC 21		LXI	H,LET	
10BD	CD 23 20		CALL	PDATA	
10C0	21 B5 21	PRMPT	LXI	H,GUESS	PRINT GUESS PROMPT
10C3	CD 2B 20		CALL	PSTRG	
10C6	CD 19 20		CALL	INCH	GET GUESS
10C9	FE 5B		CPI	'Z'+1	IS IT ABOVE Z ?
10CB	F2 C0 20		JP	PRMPT	
10CE	FE 41		CPI	'A'	IS IT BELOW A ?
10D0	FA C0 20		JM	PRMPT	
10D3	21 0A 20		LXI	H,WDBUF	POINT TO WORD BUFFER
10D6	47		MOV	B,A	PUT GUESS IN B
10D7	7E	GTTRY	MOV	A,M	GET A CHARACTER
10D8	FE 04		CPI	04	IS IT END CHARACTER
10DA	CA EB 20		JZ	CHKD	IF SO ALL DONE
10DD	B8		CMP	B	DOES THE GUESS MATCH?
10DE	CA E5 20		JZ	MATCH	
10E1	23		INX	H	POINT TO NEXT LETTER
10E2	C3 D7 20		JMP	GTTRY	
10E5	F6 80	MATCH	ORI	80H	SET MATCH INDICATOR
10E7	77		MOV	M,A	STORE IT BACK
10E8	C3 F2 20		JMP	DONE2	
10EB	3A 09 10	CHKD	LDA	HGCNT	INC. ERROR COUNTER
10EE	3C		INR	A	
10EF	32 09 20		STA	HGCNT	
10F2	CD 31 20	DONE2	CALL	PCRLF	
10F5	21 0A 20		LXI	H,WDBUF	POINT TO WORD BUFFER
10F8	7E	GETCH	MOV	A,M	GET A LETTER
10F9	B7		ORA	A	IF MATCH INDICATOR IS
10FA	FA 04 21		JM	PRINT	SET, PRINT THE LETTER
10FD	FE 04		CPI	04	IS IT THE END CHAR
10FF	CA 10 21		JZ	ENDWD	
1102	3E 2D		MVI	A,'-'	IF NOT MATCHED, PRINT -
1104	E6 7F	PRINT	ANI	7FH	STRIP OFF MATCH IND.
1106	CD 16 20		CALL	OUTCH	OUTPUT A LETTER
1109	CD 14 20		CALL	OUTS BLANK	OUTPUT A SPACE
110C	23		INX	H	
110D	C3 F8 20		JMP	GETCH	DO AGAIN
1110	CD 14 20	ENDWD	CALL	OUTS BLANK	

* LOST ROUTINE

* WON ROUTINE

* PRINT STRINGS

2275	1172	OD	00	00	CRLF	DB	ODH, OAH, O, O, O, O, 4
2279	1179	48	41	4E	47	HANG	'HANGMAN FOR 8080'
	1189	04					4 40, 41, 4E, 20, 60, 6F, 72, 38, 3D, 38, 3D
	118A	49	20	61	6D	IAM	'I AM THINKING OF A WORD'
	11A1	04					4 20, 40, 60, 6F, 72, 38, 3D, 38, 3D
	11A2	54	48	41	54		'THAT HAS'

11AB	04	DB	4
11AC	20 4C 45 54	LET DB	'LETTERS'
11B4	04	DB	4
11B5	59 4F 55 52	GUESS DB	'YOUR GUESS?'
11C1	04	DB	4
11C2	59 4F 55 20	WIN DB	'YOU WIN !!'
11CC	07 07 04	DB	7 7 4
11CF	54 48 45 20	THE DB	'THE WORD WAS'
11DC	04	DB	4
11DD	4F 55 54 20	END DB	'OUT OF WORDS!'
11EA	04	DB	4
11EB	50 4C 41 59	AGAIN DB	'PLAY AGAIN?'

* WORD LIST

11F7	04	WDLST DB	4
11F8	57 49 46 45	DB	'WIFE'
11FC	04	DB	4
11FD	52 41 44 49	DB	'RADIO'
1202	04	DB	4
1203	43 4F 4C 4C 45 47 45	DB	'COLLEGE'
120A	04	DB	4
120B	4D 49 53 55 53 45 44	DB	'MISUSED'
1212	04	DB	4
1213	5A 45 42 52 41	DB	'ZEBRA'
1218	04	DB	4
1219	43 4F 4D 50 55 54 45 52	DB	'COMPUTER'
1221	04	DB	4
1222	47 55 49 54 41 52	DB	'GUITAR'
1228	04	DB	4
1229	42 4F 54 54 4C 45	DB	'BOTTLE'
122F	04	DB	4
1230	45 58 54 52 41	DB	'EXTRA'
1235	04	DB	4
1236	46 4F 52 57 41 52 44	DB	'FORWARD'
123D	04	DB	4
123E	48 41 4C 46	DB	'HALF'
1242	04	DB	4
1243	46 4F 58	DB	'FOX'
1246	04	DB	4
1247	50 4F 44	DB	'POD'
124A	04	DB	4
124B	41 52 54 49 53 54	DB	'ARTIST'
1251	04	DB	4
1252	53 4B 49 49 4E 47	DB	'SKIING'
1258	04	DB	4
1259	53 4E 4F 57	DB	'SNOW'
125D	04	DB	4
125E	59 45 4C 4C 4F 57	DB	'YELLOW'
1264	04	DB	4
1265	42 49 43 59 43 4C 45	DB	'BICYCLE'
126C	04	DB	4
126D	53 4F 46 41	DB	'SOFA'
1271	04	DB	4

1272	48 41 4D 4D 45 52	DB	'HAMMER'
1278	04	DB	4
1279	43 48 55 52 43 48	DB	'CHURCH'
127F	04	DB	4
1280	4D 55 53 49 43	DB	'MUSIC'
1285	04	DB	4
1286	4A 41 49 4C	DB	'JAIL'
128A	04	DB	4
128B	4F 43 54 4F 50 55 53	DB	'OCTOPUS'
1292	04	DB	4
1293	43 4F 4C 4F 47 4E 45	DB	'COLOGNE'
129A	04	DB	4
129B	4B 4E 49 46 45	DB	'KNIFE'
12A0	04	DB	4
12A1	50 41 52 41 4B 45 45 54	DB	'PARAKEET'
12A9	04	DB	4
12AA	4B 4E 55 43 4B 4C 45	DB	'KNUCKLE'
12B1	04	DB	4
12B2	56 45 54 4F	DB	'VETO'
12B6	04	DB	4
12B7	48 45 58 41 47 4F 4E	DB	'HEXAGON'
12BE	04	DB	4
12BF	50 41 52 41 4C 4C 45 4C	DB	'PARALLEL'
12C7	04	DB	4
12C8	48 45 41 44 4C 49 47 48 54	DB	'HEADLIGHT' 4
12D1	04	DB	4
12D2	57 49 46 45	DB	'WIFE'
12D6	04	DB	4
12D7	52 41 44 49 4F	DB	'RADIO'
12DC	04	DB	4
12DD	CD 20 DC	DB	4
12ED	C3 24 DC	CALL WHØ	2044
		JMP WH1	

SYMBOLIC REFERENCE TABLE.

AGAIN	11EB	LET	11AC	RLOOP	1041
BEGIN	1055	LOST	1142	RNDM	1006
CHND	10EB	MATCH	10E5	START	1065
CRLF	1172	MONTR	101C	THE	11CF
DONE1	10B4	MORE	115E	TSTWD	1135
DONE2	10F2	NOPRT	112A	WDBUF	100A
END	11DD	OUTCH	1016	WDCNT	1008
ENDWD	1110	OUTS	1014	WDLST	11F7
GETCH	10F8	PCHR	101F	WIN	11C2
GETH	1121	PCRLF	1031	WON	1158
GETRN	107B	PDATA	1023		
GTNXT	10A1	PDNXT	1022		
GTRY	10D7	PRINT	1104		
GUESS	11B5	PRMPT	10C0		
HANG	1179	PRTWD	114B		
HEADR	1072	PSNXT	102A		
HGCNT	1009	PSTRG	102B		
IAM	118A	PTNXT	1091		
INCH	1019	RANDM	103A		


```

:0610000031FF1FC3551073
:0110080020C7
:101014003E20C3F981C30582C31E80CD1610237EF2
:10102400FE04C21F10C923CD3110C32310E5217261
:1010340011CD2310E1C9E5C521061006087E07AECF
:101044001717237E17772B7E177705C24110C1E14E
:10105400C92108104ECD3110217911CD2B10CD317D
:10106400100DF2721021DD11CD2B10C31C10218A3A
:1010740011CD2B10CD2A10CD3A10E63FCA7B104774
:101084003A0810B8FA7B10110A1021F611237EFEDB
:1010940004C2911005C29110AF47320910237EB7E4
:1010A400CA7B1012FE04CAB41004133600C3A11084
:1010B4003E3080CD161021AC11CD231021B511CDB9
:1010C4002B10CD1910FE5BF2C010FE41FAC01021A6
:1010D4000A10477EFE04CAEB10B8CAE51023C3D732
:1010E40010F68077C3F2103A09103C320910CD3162
:1010F40010210A107EB7FA0411FE04CA10113E2D05
:10110400E67FCD1610CD141023C3F810CD1410CDE6
:1011140014102179113A0910B7CA2A11477ECD1645
:10112400102305C221113A0910FE07CA4211210AEF
:10113400107EFE04CA5811F2C01023C3351121CF0A
:1011440011CD2B10210A107EFE04CA5E11CD16109B
:1011540023C34B1121C211CD2B10CD311021EB1122
:10116400CD2B10CD1910FE4ECA1C10C365100D0AEC
:101174000000000000448414E474D414E20464F5266
:101184002038303830044920414D205448494E4BD2
:10119400494E47204F46204120574F52440454485B
:1011A4004154204841532004204C45545445525343
:1011B40004594F55522047554553533F2004594F26
:1011C400552057494E2021210707045448452057EC
:1011D4004F52442057415320044F5554204F46202A
:1011E400574F5244532104504C41592041474149DF
:1011F4004E3F20045749464504524144494F044355
:101204004F4C4C454745044D495355534544045AA6
:101214004542524104434F4D50555445520447559D
:101224004954415204424F54544C45044558545275
:101234004104464F52574152440448414C460446E7
:101244004F5804504F440441525449535404534B8F
:1012540049494E4704534E4F570459454C4C4F5738
:101264000442494359434C4504534F4641044841C1
:101274004D4D455204434855524348044D55534936
:1012840043044A41494C044F43544F50555304437B
:101294004F4C4F474E45044B4E494645045041522E
:1012A400414B454554044B4E55434B4C4504564520
:1012B400544F0448455841474F4E04504152414C05
:0E12C4004C454C04484541444C4947485404AD
**
**
:00000000000

```


HANGMAN
 I AM THINKING OF A WORD
 THAT HAS 6 LETTERS
 YOUR GUESS? E
 - - - - E -
 YOUR GUESS? I
 - - - - E - H
 YOUR GUESS? A
 - - - - E - HA
 YOUR GUESS? O
 - - - - E - HAN
 YOUR GUESS? U
 - U - - E - HAN
 YOUR GUESS? T
 T U - - E - HAN
 YOUR GUESS? L
 T U - - E - HANG
 YOUR GUESS? R
 T U R - E - HANG
 YOUR GUESS? Y
 T U R - E Y HANG
 YOUR GUESS? K
 T U R K E Y
 WANT TO PLAY AGAIN? Y

HANGMAN
 I AM THINKING OF A WORD
 THAT HAS 6 LETTERS
 YOUR GUESS? E
 - - - - E -
 YOUR GUESS? A
 - A - - E -
 YOUR GUESS? I
 - A - - E - H
 YOUR GUESS? R
 - A - - E R H
 YOUR GUESS? H
 - A - H E R H
 YOUR GUESS? T
 - A T H E R H
 YOUR GUESS? L
 - A T H E R HA
 YOUR GUESS? G
 - A T H E R HAN
 YOUR GUESS? F
 F A T H E R
 WANT TO PLAY AGAIN? Y

HANGMAN
 I AM THINKING OF A WORD
 THAT HAS 5 LETTERS
 YOUR GUESS? E
 - - - E -
 YOUR GUESS? I
 - - - E - H
 YOUR GUESS? A
 - - - E - HA
 YOUR GUESS? O
 - O - E - HA
 YOUR GUESS? R
 - O - E - HAN
 YOUR GUESS? T
 - O - E - HANG
 YOUR GUESS? B
 B O - E - HANG
 YOUR GUESS? P
 B O - E - HANGM
 YOUR GUESS? M
 B O - E - HANGMA
 YOUR GUESS? D
 B O - E - HANGMAN
 THE WORD WAS BOXES
 WANT TO PLAY AGAIN? Y

HANGMAN
 I AM THINKING OF A WORD
 THAT HAS 6 LETTERS
 YOUR GUESS? E
 - - - - E
 YOUR GUESS? A
 - - A - - E
 YOUR GUESS? I
 - - A - - E H
 YOUR GUESS? O
 - - A - - E HA
 YOUR GUESS? U
 - - A - - E HAN
 YOUR GUESS? S
 S - A - - E HAN
 YOUR GUESS? H
 S - A - - E HANG
 YOUR GUESS? T
 S T A - - E HANG
 YOUR GUESS? T
 S T A - - E HANGM
 YOUR GUESS? B
 S T A - - E HANGMA
 YOUR GUESS? L
 S T A - L E HANGMA
 YOUR GUESS? P
 S T A P L E
 WANT TO PLAY AGAIN? N

*
* HURKLE FOR 8080
*
*
* COPYRIGHT (C) 1977 BY
* TECHNICAL SYSTEMS CONSULTANTS
* BOX 2574 W. LAFAYETTE, IN. 47906
*
* THIS GAME IS ADAPTED FOR THE 8080 FROM THE
* VERSION RELEASED BY PEOPLES COMPUTER CO.
* THE HURKLE IS A WILY CREATURE WHO TAKES
* GREAT PLEASURE IN HIDING FROM HUNTERS.
* YOUR TASK, AS HEAD HURKLE HUNTER, IS TO
* FIND THE HIDING HURKLE IN 3 GUESSES OR
* LESS. HE WILL BE HIDING ON A 10X10 GRID
* WITH 0,0 BEING THE SOUTHWEST CORNER. UPON
* STARTING THE GAME, YOU WILL BE ASKED FOR
* AN INITIAL GUESS. YOU SHOULD RESPOND BY
* GUESSING A NORTH-SOUTH COORDINATE (0-9),
* FOLLOWED BY AN EAST-WEST COORDINATE (0-9).
* INCREASING VALUES MOVE YOU NORTH AND EAST.
* IF HE WAS NOT THERE, THE COMPUTER WILL
* GIVE YOU A CLUE, TELLING YOU IN WHICH
* DIRECTION YOU NEED TO GO TO FIND HIM. FOR
* EXAMPLE, IF YOU GUESSED 2,3 AND THE HURKLE
* WAS HIDING AT 6,6, YOU WOULD BE TOLD TO
* GO NORTHEAST. IF YOU CAN'T FIND HIM IN
* 3 GUESSES, YOU WILL BE TOLD WHERE HE WAS
* HIDING.
*
* SEE ATTACHED SAMPLE OUTPUT FOR AN EXAMPLE.
*
* THIS PROGRAM ASSUMES THAT THE USER HAS AN
* I/O TERMINAL OF SOME TYPE AND ROUTINES TO
* PRINT AN ASCII CHARACTER FROM THE A REGISTER
* AND INPUT AN ASCII CHARACTER INTO THE A
* REGISTER. THE USER MUST LOAD THE ADDRESS
* OF THE OUTPUT ROUTINE AT LOCATION 100B HEX
* AND THE ADDRESS OF THE INPUT ROUTINE AT
* 100E HEX (IN NORMAL 8080 FORM). ALSO, THE
* USER MUST INSTALL THE ENTRY ADDRESS OF HIS
* MONITOR AT 1011 HEX. THE SOURCE LISTING
* IS SUPPLIED WITH THE ADDRESSES SET FOR THE
* TSC 8080 MONITOR. THE STACK ADDRESS IS
* SPECIFIED AT LOCATION 1001 HEX. IF IT IS
* NECESSARY THIS ADDRESS CAN BE MODIFIED TO
* SUIT YOUR PARTICULAR SYSTEM.
*
* THE STARTING ADDRESS OF THIS PROGRAM IS
* 1000 HEX. WHEN THE COMPUTER ASKS TO PLAY
* AGAIN, TYPE Y FOR YES OR N FOR NO.
* CAUTION MUST BE EXERCISED THAT BYTES RNDM

* AND RNDM+1 (LOCATIONS 1006 AND 1007 HEX)
 * ARE NOT BOTH ZERO. IF THIS IS THE CASE,
 * CHANGE ONE OR BOTH BYTES TO SOME NON-ZERO
 * VALUE BEFORE RUNNING THE PROGRAM.

*

*

* HAPPY HURKLE HUNTING !!!

*

1000 *00 00 00* ORG 1000H
 1000 *31 00 0F* LXI S,1FFFH SETUP STACK
 1003 C3 49 10 *0C* JMP BEGIN START THE GAME
 1006 *03 AC* RNDM DS 2
 1008 *00* YSTOR DS 1
 1009 *00* XSTOR DS 1

* JUMP TO EXTERNAL ROUTINES

100A C3 *24 0C* ~~52 00~~ OUTCH JMP 081F9H
 100D C3 *05 00 20 0C* ~~05 00 20 0C~~ INCH JMP 08205H *C3 60 0C*
 1010 C3 *57 00* ~~57 00~~ MONTR JMP 0801EH
00 00

*0C60 CD200C
 C3 240C
 MAKE SURE
 1006,1007 are not 0*

* PRINT ROUTINES

0C13 1013 CD 0A 10 *20* PCHR CALL OUTCH PRINT CHARACTER
 1016 23 PDNXT INX H
 1017 7E PDATA MOV A,M GET CHARACTER
 1018 FE 04 CPI 4 CHECK FOR EOT
 101A C2 13 10 *20* JNZ PCHR
 101D C9 RET
 101E 23 PSNXT INX H POINT TO NEXT STRING
 101F CD 25 10 *20* PSTRG CALL PCRLF PRINT CR AND LF
 1022 C3 17 10 *20* JMP PDATA PRINT STRING
0C25 1025 E5 PCRLF PUSH H SAVE H AND L
 1026 21 *44 11 22* LXI H,CRLF
 1029 CD 17 10 *20* CALL PDATA PRINT CR AND LF
 102C E1 POP H RESTORE H AND L
 102D C9 RET

* RANDOM NUMBER GENERATOR

0C2E 102E E5 RANDM PUSH H SAVE H AND L
 102F C5 PUSH B SAVE B REGISTER
 1030 21 06 10 *20* LXI H,RNDM POINT TO RNDM
 1033 06 08 MVI B,B SET LOOP COUNTER
0C35 1035 7E RLOOP MOV A,M GET MS BYTE OF RNDM
 1036 07 RLC ALIGN BITS 13 AND 14
 1037 AE XRA M XOR WITH RNDM
 1038 17 RAL GET BIT 13.XOR.14
 1039 17 RAL INTO CARRY BIT
 103A 23 INX H
 103B 7E MOV A,M ROTATE BOTH
 103C 17 RAL BYTES OF RNDM
 103D 77 MOV M,A PUTTING CARRY
 103E 2B DCX H INTO LSB

103F	7E	MOV	A,M	
1040	17	RAL		
1041	77	MOV	M,A	
1042	05	DCR	B	
1043	C2 35 10 20	JNZ	RLOOP	DO LOOP 8 TIMES
1046	C1	POP	B	RESTORE B REGISTER
1047	E1	POP	H	RESTORE H AND L
1048	C9	RET		

* MAIN PROGRAM

1049	CD 2E 10 20	BEGIN	CALL	RANDM	GET A RANDOM NUMBER
104C	E6 0F		ANI	0FH	MASK IT
104E	FE 0A		CPI	0AH	BE SURE IT'S UNDER 10
1050	F2 49 10 20		JP	BEGIN	
1053	F6 30		ORI	30H	MAKE IT ASCII
1055	32 08 10 20		STA	YSTOR	SAVE IT AS Y
1058	CD 2E 10 20	GETX	CALL	RANDM	GET A RANDOM NUMBER
105B	E6 0F		ANI	0FH	MASK IT
105D	FE 0A		CPI	0AH	BE SURE IT'S UNDER 10
105F	F2 58 10 20		JP	GETX	
1062	F6 30		ORI	30H	MAKE IT ASCII
1064	32 09 10 20		STA	XSTOR	SAVE IT AS X
1067	0E 31		MVI	C,31H	SET NO. OF TRIES TO 1
1069	CD 25 10 20		CALL	PCRLF	
106C	21 65 11 22		LXI	H,INTRO	
106F	CD 1F 10 20		CALL	PSTRG	OUTPUT INTRO
1072	21 7B 11 22	LOOP	LXI	H,GUESS	
0075 1075	CD 1F 10 20		CALL	PSTRG	OUTPUT GUESS NUMBER
1078	79		MOV	A,C	
1079	CD 0A 10 20		CALL	OUTCH	
107C	CD 16 10 20		CALL	PDXNT	
107F	CD 0D 10 20		CALL	INCH	GET GUESS FOR Y
1082	FE 30		CPI	30H	IS IT VALID?
1084	FA 8C 10 20		JM	NTVLD	
1087	FE 3A		CPI	3AH	
1089	FA 95 10 20		JM	CONT1	IF SO, CONTINUE
108C	21 F5 11 22	NTVLD	LXI	H,DUMB	ELSE REPORT ERROR
108F	CD 1F 10 20		CALL	PSTRG	
1092	C3 72 10 20		JMP	LOOP	
1095	57	CONT1	MOV	D,A	SAVE Y GUESS IN D
1096	21 83 11 22		LXI	H,STR1	
1099	CD 17 10 20		CALL	PDATA	
109C	CD 0D 10 20		CALL	INCH	GET GUESS FOR X
109F	FE 30		CPI	30H	IS IT VALID?
10A1	FA 8C 10 20		JM	NTVLD	
10A4	FE 3A		CPI	3AH	
10A6	F2 8C 10 20		JP	NTVLD	IF NOT GO REPORT ERROR
10A9	5F		MOV	E,A	ELSE SAVE X GUESS IN E
10AA	3A 08 10 20		LDA	YSTOR	
10AD	BA		CMP	D	COMPARE GUESS TO Y LOC.
10AE	C2 B6 10 20		JNZ	CONT2	IF NOT EQUAL, CONTINUE
10B1	16 FF		MVI	D,OFFH	ELSE MAKE D NEGATIVE
10B3	C3 D4 10 20		JMP	CHKX	

10B6	F2 C8 20	CONT2	JP	GON	IF GUESS IS LOW, GON
10B9	21 4B 22		LXI	H,GOSTR	
10BC	CD 1F 20		CALL	PSTRG	ELSE,
10BF	21 55 22		LXI	H,SOUTH	OUTPUT 'GO SOUTH'
10C2	CD 17 20		CALL	PDATA	
10C5	C3 D4 20		JMP	CHKX	
10C8	21 4B 22	GON	LXI	H,GOSTR	
10CB	CD 1F 20		CALL	PSTRG	
10CE	21 4F 22		LXI	H,NORTH	OUTPUT 'GO NORTH'
10D1	CD 17 20		CALL	PDATA	
10D4	AF	CHKX	XRA	A	A NEGATIVE D SHOWS
10D5	82		ADD	D	Y GUESS WAS RIGHT
10D6	F2 F6 20		JP	NOTEQ	SO, JUMP TO NOTEQ IF +
10D9	3A 09 20		LDA	XSTOR	
10DC	BB		CMP	E	COMPARE GUESS TO X LOC.
10DD	C2 F0 20		JNZ	PRTGO	IF NOT EQUAL PRINT GO
10E0	21 87 22		LXI	H,FOUND	ELSE, YOU FOUND HIM
10E3	CD 1F 20		CALL	PSTRG	OUTPUT IT
10E6	79		MOV	A,C	
10E7	CD 0A 20		CALL	OUTCH	OUTPUT NUMBER OF TRIES
10EA	CD 16 20		CALL	PDXNT	
10ED	C3 30 22		JMP	PLAGN	PLAY AGAIN?
10F0	21 4B 22	PRTGO	LXI	H,GOSTR	
10F3	CD 1F 20		CALL	PSTRG	PRINT 'GO'
10F6	3A 09 20	NOTEQ	LDA	XSTOR	
10F9	BB		CMP	E	COMPARE GUESS TO X LOC.
10FA	CA 0F 22		JZ	NTFND	
10FD	F2 09 22		JP	GOE	IF GUESS IS LOW, GOE
1100	21 60 22		LXI	H,WEST	ELSE, MUST GO WEST
1103	CD 17 20		CALL	PDATA	
1106	C3 0F 22		JMP	NTFND	
1109	21 5B 22	GOE	LXI	H,EAST	MUST GO EAST
110C	CD 17 20		CALL	PDATA	
110F	0C	NTFND	INR	C	DID NOT FIND HIM
1110	79		MOV	A,C	INCREMENT NO. OF TRIES
1111	FE 34		CPI	34H	ANY TRIES LEFT?
1113	C2 72 20		JNZ	LOOP	IF SO, CONTINUE
1116	21 C7 22		LXI	H,LOST	ELSE REPORT LOSS
1119	CD 1F 20		CALL	PSTRG	
111C	CD 1E 20		CALL	PSNXT	
111F	3A 08 20		LDA	YSTOR	
1122	CD 0A 20		CALL	OUTCH	OUTPUT HIS Y LOC.
1125	3E 2C		MVI	A,','	
1127	CD 0A 20		CALL	OUTCH	OUTPUT A COMMA
112A	3A 09 20		LDA	XSTOR	
112D	CD 0A 20		CALL	OUTCH	OUTPUT HIS X LOC.
1130	21 AB 22	PLAGN	LXI	H,AGAIN	
1133	CD 25 20		CALL	PCRLF	
1136	CD 1F 20		CALL	PSTRG	PLAY AGAIN?
1139	CD 0D 20		CALL	INCH	INPUT Y OR N
113C	FE 4E		CPI	'N'	N MEANS ALL DONE
113E	CA 10 20		JZ	MONTR	SO EXIT PROGRAM
1141	C3 49 20		JMP	BEGIN	ELSE, START NEW GAME

1147 00 04 00 00

* PRINT STRINGS

1144 0B 04 00 00 CRLF DB ODH, OAH, 0, 0, 0, 0, 4
 114B 47 4F 20 GOSTR DB 'GO '
 114E 04 DB 4
 114F 4E 4F 52 54 48 NORTH DB 'NORTH'
 1154 04 DB 4
 1155 53 4F 55 54 48 SOUTH DB 'SOUTH'
 115A 04 DB 4
 115B 45 41 53 54 EAST DB 'EAST'
 115F 04 DB 4
 1160 57 45 53 54 WEST DB 'WEST'
 1164 04 DB 4
 1165 54 48 45 20 48 55 52 40 4C DB 20 49 53 20 48 49 44 49 4E 47 21
 117A 04 DB 4
 117B 47 55 45 53 53 20 23 GUESS DB 'GUESS #'
 1182 04 DB 4
 1183 20 3F 20 STR1 DB ' ? '
 1186 04 DB 4
 1187 59 4F 55 20 46 4F 55 4E 44 20 48 49 40 20 48 4E 20 FOUND DB 'YOU FOUND HIM IN '
 1198 04 DB 4
 1199 20 47 55 45 53 55 45 53 2E 20 21 21 DB 4
 11A6 07 04 DB 7, 4
 11A8 57 4F 55 4C 44 AGAIN DB 'WOULD YOU LIKE TO PLAY AGAIN? '
 11C6 04 DB 4
 11C7 53 4F 52 52 59 LOST DB 'SORRY, THAT WAS 3 GUESSES.'
 11E1 04 DB 4
 11E2 54 48 45 20 DB 'THE HURKLE WAS HIDING AT '
 11FB 04 DB 4
 11FC 59 4F 55 20 DUMB DB 'YOU GOOFED, TRY AGAIN.'
 1212 04 DB 4
 1213 END DB 54, 52, 59, 20, 41, 47, 41, 49, 4E, 2E, 04

SYMBOLIC REFERENCE TABLE.

AGAIN	11A8	GUESS	117B	PDATA	1017
BEGIN	1049	INCH	100D	PDNXT	1016
CHKX	10D4	INTRO	1165	PLAGN	1130
CONT1	1095	LOOP	1072	PRTGO	10F0
CONT2	10B6	LOST	11C7	PSNXT	101E
CRLF	1144	MONTR	1010	PSTRG	101F
DUMB	11FC	NORTH	114F	RANDM	102E
EAST	115B	NOTEQ	10F6	RLOOP	1035
FOUND	1187	NTFND	110F	RNDM	1006
GETX	1058	NTVLD	108C	SOUTH	1155
GOE	1109	OUTCH	100A	STR1	1183
GON	10C8	PCHR	1013	WEST	1160
GOSTR	114B	PCRLF	1025	XSTOR	1009
				YSTOR	1008

HURKLE
2000, 2210

MAKE SURE RNDM
CONTAINS NON ZERO
PUT AT 060 CD 2000, 03 24 DC
PUT AT 2200 F2

* 0000000000

THE HURKLE IS HIDING!
 GUESS #1 ? 5 ? 5
 GO WEST
 GUESS #2 ? 5 ? 3
 YOU FOUND HIM IN 2 GUESSES !!!
 WANT TO PLAY AGAIN? Y

THE HURKLE IS HIDING!
 GUESS #1 ? 5 ? 5
 GO SOUTHEAST
 GUESS #2 ? 3 ? 8
 YOU FOUND HIM IN 2 GUESSES !!!
 WANT TO PLAY AGAIN? Y

THE HURKLE IS HIDING!
 GUESS #1 ? 5 ? 5
 GO SOUTHEAST
 GUESS #2 ? 3 ? 8
 GO EAST
 GUESS #3 ? 3 ? 9
 YOU FOUND HIM IN 3 GUESSES !!!
 WANT TO PLAY AGAIN? Y

THE HURKLE IS HIDING!
 GUESS #1 ? 5 ? 5
 GO WEST
 GUESS #2 ? 5 ? 3
 YOU FOUND HIM IN 2 GUESSES !!!
 WANT TO PLAY AGAIN? Y

THE HURKLE IS HIDING!
 GUESS #1 ? 5 ? 5
 GO NORTHWEST
 GUESS #2 ? 8 ? 3
 GO WEST
 GUESS #3 ? 8 ? 2
 GO WEST
 SORRY, THAT WAS 3 GUESSES.
 THE HURKLE WAS AT 8.0
 WANT TO PLAY AGAIN? Y

THE HURKLE IS HIDING!
 GUESS #1 ? 5 ? 5
 GO SOUTHWEST
 GUESS #2 ? 3 ? 3
 GO SOUTH
 GUESS #3 ? 1 ? 3
 GO SOUTH
 SORRY, THAT WAS 3 GUESSES.
 THE HURKLE WAS AT 0.3
 WANT TO PLAY AGAIN? Y

THE HURKLE IS HIDING!
 GUESS #1 ? 5 ? 5
 GO NORTHWEST
 GUESS #2 ? 8 ? 2
 GO EAST
 GUESS #3 ? 8 ? 4
 GO WEST
 SORRY, THAT WAS 3 GUESSES.
 THE HURKLE WAS AT 8.3
 WANT TO PLAY AGAIN? Y

THE HURKLE IS HIDING!
 GUESS #1 ? 5 ? 5
 GO SOUTHWEST
 GUESS #2 ? 3 ? 3
 GO WEST
 GUESS #3 ? 3 ? 1
 GO EAST
 SORRY, THAT WAS 3 GUESSES.
 THE HURKLE WAS AT 3.2
 WANT TO PLAY AGAIN? N

MASTERMIND 8080

COPYRIGHT (C) 1976 BY

TECHNICAL SYSTEMS CONSULTANTS
BOX 2574 W. LAFAYETTE IN 47906
(317) 742-7509

* THIS PROGRAM IMPLEMENTS THE POPULAR MASTERMIND
* GAME FOR 8080 BASED COMPUTERS. THE PROGRAM
* ASSUMES THAT THE USER HAS AN I/O TERMINAL OF SOME
* TYPE AND A ROUTINE FOR PRINTING AN ASCII CHARACTER
* AND INPUTTING AN ASCII CHARACTER USING THAT DEVICE.
* THE USER MUST INSTALL THE ADDRESSES OF THOSE I/O
* ROUTINES AT 1013 AND 1016 (IN NORMAL 8080 FORM)
* FOR OUTPUT AND INPUT RESPECTIVELY. ALSO, THE USER
* MUST INSTALL THE ENTRY ADDRESS OF HIS MONITOR
* AT 1019. THE SOURCE LISTING IS SUPPLIED WITH THE
* ADDRESSES SET FOR THE TSC 8080 MONITOR. THE STACK
* ADDRESS IS SPECIFIED AT LOCATION 1000. YOU SHOULD
* MODIFY THIS TO SUIT YOUR SYSTEM.
* THE OBJECT OF THIS GAME IS TO GUESS A SEQUENCE
* OF 4 LETTERS IN THE RANGE OF A-F THAT THE COMPUTER
* HAS SELECTED. THE COMPUTER WILL GIVE YOU CLUES
* AS TO THE ACCURACY OF YOUR GUESS AS FOLLOWS:
* ONE BLACK MARKER FOR EACH LETTER GUESSED IN THE
* CORRECT POSITION.
* ONE WHITE MARKER FOR EACH LETTER GUESSED IN THE
* SEQUENCE BUT NOT IN THE CORRECT POSITION.
* THE TOTAL NUMBER OF MARKERS (BLACK + WHITE) WILL
* NEVER EXCEED 4 BECAUSE THE COMPUTER WILL AWARD AT
* MOST ONE MARKER FOR EACH POSITION OF THE USER'S
* GUESS. WHEN YOU RECEIVE 4 BLACK MARKERS YOU HAVE
* GUESSED THE SEQUENCE AND THE COMPUTER WILL INFORM
* YOU AS TO HOW MANY TRIES YOU NEEDED.
* IF YOU GIVE UP, YOU MAY TYPE A "G" AS ONE OF
* OF YOUR GUESSES AND THE COMPUTER WILL TELL YOU
* WHAT THE SEQUENCE WAS.
* SEE THE ATTACHED SAMPLE OUTPUT FOR AN EXAMPLE.
* CAUTION MUST BE EXERCISED SO THAT NOT BOTH BYTES
* RNDM AND RNDM+1 ARE ZERO. IF THIS IS THE CASE,
* CHANGE ONE OR BOTH BYTES TO SOME VALUE (NON-ZERO)
* BEFORE YOU BEGIN RUNNING THE PROGRAM.
* THE STARTING ADDRESS OF THIS PROGRAM IS 1000.

HAVE FUN!

1000		ORG	1000H	
1000	31 00 01	START LXI	S,100H	INITIALIZE SP


```

1003      C3 4D 10      JMP      BEGIN      GO TO PROGRAM
          *
          *
1006      RNDM      DS      2
1008      TOKENS      DS      4
100C      HIS      DS      4
          *
          *
1010      3E 20      OUTS      MVI      A, ' '
1012      C3 F9 81      OUTCH      JMP      81F9H
1015      C3 05 82      INCH      JMP      8205H
1018      C3 1E 80      MON      JMP      801EH
101B      21 69 11      PCRLF      LXI      H, CRLF
101E      7E          PDATA      MOV      A, M
101F      FE 04          CPI      4
1021      C8          RZ
1022      CD 12 10      CALL      OUTCH
1025      23          PDNXT      INX      H
1026      C3 1E 10      JMP      PDATA
1029      23          PNXT      INX      H      BUMP TEXT POINTER
102A      E5          PSTR      PUSH     H
102B      CD 1B 10      CALL      PCRLF
102E      E1          POP      H
102F      C3 1E 10      JMP      PDATA
1032      E5          RANDOM      PUSH     H
1033      C5          PUSH     B
1034      21 06 10      LXI      H, RNDM
1037      06 08          MVI      B, 8
1039      7E          RLOOP      MOV      A, M
103A      07          RLC
103B      AE          XRA      M
103C      17          RAL
103D      17          RAL
103E      23          INX      H
103F      7E          MOV      A, M
1040      17          RAL
1041      77          MOV      M, A
1042      2B          DCX      H
1043      7E          MOV      A, M
1044      17          RAL
1045      77          MOV      M, A
1046      05          DCR      B
1047      C2 39 10      JNZ      RLOOP
104A      C1          POP      B
104B      E1          POP      H
104C      C9          RET
104D      CD 1B 10      BEGIN      CALL      PCRLF
1050      CD 29 10      CALL      PNXT
1053      CD 29 10      CALL      PNXT
1056      CD 29 10      CALL      PNXT
1059      11 08 10      LXI      D, TOKENS      SET INDEX POINTER
105C      06 04          MVI      B, 4      SET COUNTER
105E      CD 32 10      SELECT      CALL      RANDOM      GET NUMBER
1061      E6 07          ANI      07      MASK TO 0-7

```


1063	FE 06		CPI	06	CHECK IN RANGE
1065	D2 5E 10		JNC	SELECT	IF NOT GET ANOTHER
1068	C6 41		ADI	41H	ADD ON ASCII BIAS
106A	12		STAX	D	STORE
106B	13		INX	D	ADVANCE PTR
106C	05		DCR	B	CHECK DONE
106D	C2 5E 10		JNZ	SELECT	
1070	AF		XRA	A	CLEAR A
1071	F5		PUSH	PSW	SAVE
1072	06 04	GETTRY	MVI	B,4	SET FOR 4
1074	21 08 10		LXI	H,TOKENS	SET POINTER
1077	7E	CLRFLG	MOV	A,M	GET TOKEN
1078	E6 4F		ANI	4FH	CLEAR OUT FLAGS
107A	77		MOV	M,A	RESTORE
107B	23		INX	H	ADV PTR
107C	05		DCR	B	CHECK DONE
107D	C2 77 10		JNZ	CLRFLG	
1080	06 04		MVI	B,4	SET COUNTER
1082	21 AB 11		LXI	H,GUESS	
1085	CD 2A 10		CALL	PSTR	PRINT MESSAGE
1088	21 0C 10		LXI	H,HIS	POINT TO STORE
108B	CD 15 10	LOOP1	CALL	INCH	GET A CHAR
108E	FE 41		CPI	'A'	IS IT <A
1090	FA 98 10		JM	ERROR	
1093	FE 48		CPI	'G'+1	IS IT >G
1095	DA A5 10		JC	OK	
1098	3E 3F	ERROR	MVI	A,'?'	
109A	CD 12 10		CALL	OUTCH	PRINT ?
109D	3E 07		MVI	A,7	
109F	CD 12 10		CALL	OUTCH	RING BELL
10A2	C3 72 10		JMP	GETTRY	RECYCLE
10A5	FE 47	OK	CPI	'G'	CHECK FOR GIVE UP
10A7	CA 4F 11		JZ	GIVEUP	
10AA	77		MOV	M,A	SAVE GUESS
10AB	CD 10 10		CALL	OUTS	PRINT SPACE
10AE	23		INX	H	ADVANCE PTR
10AF	05		DCR	B	CHECK DONE
10B0	C2 8B 10		JNZ	LOOP1	
10B3	F1		POP	PSW	RESTORE PSW
10B4	C6 01		ADI	01	INCREMENT TRIES
10B6	27		DAA		
10B7	F5		PUSH	PSW	SAVE
10B8	11 08 10		LXI	D,TOKENS	RESET POINTER
10BB	21 0C 10		LXI	H,HIS	POINT TO HIS
10BE	06 00		MVI	B,0	SET COUNTER
10C0	0E 04		MVI	C,4	SET INDEX
10C2	1A	LOOP2	LDAX	D	GET HIS TRY
10C3	BE		CMP	M	CHECK MATCH
10C4	C2 CE 10		JNZ	CHK1	
10C7	F6 80		ORI	80H	SET FLAG
10C9	12		STAX	D	RESTORE MARKED TOKEN
10CA	F6 20		ORI	20H	SET OTHER FLAG
10CC	77		MOV	M,A	RESTORE MARKED
10CD	04		INR	B	INDICATE BLACK

10CE	13	CHK1	INX	D	
10CF	23		INX	H	ADVANCE POINTERS
10D0	0D		DCR	C	CHECK DONE
10D1	C2 C2 10		JNZ	LOOP2	
10D4	CD 10 10		CALL	OUTS	PRINT SPACE
10D7	C5		PUSH	B	SAVE BLACKS
10D8	78		MOV	A,B	GET BLACK COUNT
10D9	F6 30		ORI	30H	MAKE ASCII
10DB	CD 12 10		CALL	OUTCH	PRINT IT
10DE	21 C0 11		LXI	H,BLKS	POINT TO TEXT
10E1	CD 1E 10		CALL	PDATA	PRINT IT
10E4	11 0C 10		LXI	D,HIS	SET INDEX
10E7	3E 04		MVI	A,4	
10E9	06 00		MVI	B,0	SET COUNTER
10EB	F5	LOOP3	PUSH	PSW	SAVE COUNTER
10EC	21 08 10	SCAN	LXI	H,TOKENS	SET INDEX
10EF	0E 04		MVI	C,4	SET COUNTER
10F1	1A	LOOP4	LDAX	D	GET TRY
10F2	BE		CMF	M	CHECK MATCH
10F3	CA 04 11		JZ	MATCH	
10F6	23		INX	H	BUMP
10F7	0D		DCR	C	DONE?
10F8	C2 F1 10		JNZ	LOOP4	
10FB	13	LOOP5	INX	D	BUMP POINTER
10FC	F1		POP	PSW	GET COUNTER
10FD	3D		DCR	A	CHECK DONE
10FE	C2 EB 10		JNZ	LOOP3	
1101	C3 0B 11		JMP	PRTWHT	
1104	F6 80	MATCH	ORI	80H	SET FLAG
1106	77		MOV	M,A	RESTORE
1107	04		INR	B	KICK COUNTER
1108	C3 FB 10		JMP	LOOP5	
110B	78	PRTWHT	MOV	A,B	GET COUNT
110C	F6 30		ORI	30H	MAKE ASCII
110E	CD 12 10		CALL	OUTCH	PRINT IT
1111	21 C7 11		LXI	H,WHTS	
1114	CD 1E 10		CALL	PDATA	PRINT TITLE
1117	F1		POP	PSW	GET BLACK COUNT
1118	FE 04		CPI	4	CHECK WON
111A	C2 72 10		JNZ	GETTRY	
111D	21 CC 11		LXI	H,YOU	POINT TO TEXT
1120	CD 2A 10		CALL	PSTR	
1123	F1		POP	PSW	GET TRY COUNTER
1124	47		MOV	B,A	SAVE
1125	E6 F0		ANI	0F0H	MASK OUT
1127	CA 33 11		JZ	NOMS	CHECK FOR MS DIGIT
112A	0F		RRC		
112B	0F		RRC		
112C	0F		RRC		
112D	0F		RRC		MAKE LS DIGIT
112E	F6 30		ORI	30H	MAKE ASCII
1130	CD 12 10		CALL	OUTCH	PRINT IT
1133	78	NOMS	MOV	A,B	RETRIEVE
1134	E6 0F		ANI	0FH	MASK

1136	F6 30		ORI	30H	MAKE ASCII
1138	CD 12 10		CALL	OUTCH	PRINT
113B	CD 25 10		CALL	PDNXT	REST OF MESSAGE
113E	21 E0 11	PLAGN	LXI	H,PLA	
1141	CD 2A 10		CALL	PSTR	PRINT MESSAGE
1144	CD 15 10		CALL	INCH	GET RESPONSE
1147	FE 4E		CPI	'N'	CHECK NO
1149	C2 4D 10		JNZ	BEGIN	IF NOT NO, YES
114C	C3 18 10		JMP	MON	
114F	21 ED 11	GIVEUP	LXI	H,ITWAS	MESSAGE
1152	CD 2A 10		CALL	PSTR	
1155	11 08 10		LXI	D,TOKENS	SET INDEX
1158	06 04		MVI	B,4	SET COUNTER
115A	1A	GLOOP	LDAX	D	GET TOKEN
115B	CD 12 10		CALL	OUTCH	PRINT
115E	CD 10 10		CALL	OUTS	SPACE
1161	13		INX	D	BUMP POINTER
1162	05		DCR	B	CHECK DONE
1163	C2 5A 11		JNZ	GLOOP	
1166	C3 3E 11		JMP	PLAGN	
1169	0D 0A 00 00	CRLF	DB	0DH,0AH,0,0,0,0,4	
1170	4D 41 53 54		DB	'MASTERMIND 8080'	
117F	04		DB	4	
1180	49 20 41 4D		DB	'I AM THINKING OF 4 LETTERS'	
119A	04		DB	4	
119B	42 45 54 57		DB	'BETWEEN A AND F'	
11AA	04		DB	4	
11AB	57 48 41 54	GUESS	DB	'WHAT IS YOUR GUESS? '	
11BF	04		DB	4	
11C0	20 42 4C 4B	BLKS	DB	' BLK '	
11C6	04		DB	4	
11C7	20 57 48 54	WHTS	DB	' WHT '	
11CB	04		DB	4	
11CC	59 4F 55 20	YOU	DB	'YOU WIN IN '	
11D7	04		DB	4	
11D8	20 54 52 49		DB	' TRIES.'	
11DF	04		DB	4	
11E0	50 4C 41 59	PLA	DB	'PLAY AGAIN? '	
11EC	04		DB	4	
11ED	49 54 20 57	ITWAS	DB	'IT WAS '	
11F4	04		DB	4	
11F5			END		

SYMBOLIC REFERENCE TABLE.

BEGIN	104D	HIS	100C	NOMS	1133	PRTWHT	110B
BLKS	11C0	INCH	1015	OK	10A5	PSTR	102A
CHK1	10CE	ITWAS	11ED	OUTCH	1012	RANDOM	1032
CLRFLG	1077	LOOP1	108B	OUTS	1010	RLOOP	1039
CRLF	1169	LOOP2	10C2	PCRLF	101B	RNDM	1006
ERROR	1098	LOOP3	10EB	PDATA	101E	SCAN	10EC
GETTRY	1072	LOOP4	10F1	PDNXT	1025	SELECT	105E
GIVEUP	114F	LOOP5	10FB	PLA	11E0	START	1000
GLOOP	115A	MATCH	1104	PLAGN	113E	TOKENS	1008
GUESS	11AB	MON	1018	PNXT	1029	WHTS	11C7
						YOU	11CC

[illegible]

MASTERMIND

I AM THINKING OF 4 LETTERS
BETWEEN A AND F

YOUR GUESS?	A A A A	1 BLK	0 WHT
YOUR GUESS?	A B B B	0 BLK	1 WHT
YOUR GUESS?	C A C C	0 BLK	1 WHT
YOUR GUESS?	D D A D	3 BLK	0 WHT
YOUR GUESS?	D D A E	2 BLK	2 WHT
YOUR GUESS?	E D A D	4 BLK	0 WHT

YOU WIN IN 6 TRIES!

PLAY AGAIN? Y

MASTERMIND

I AM THINKING OF 4 LETTERS
BETWEEN A AND F

YOUR GUESS?	A A A A	1 BLK	0 WHT
YOUR GUESS?	A B B B	0 BLK	1 WHT
YOUR GUESS?	C A C C	2 BLK	0 WHT
YOUR GUESS?	C A D D	2 BLK	2 WHT
YOUR GUESS?	A C D D	1 BLK	3 WHT

YOUR GUESS? G

IT WAS D A D C

PLAY AGAIN? Y

MASTERMIND

I AM THINKING OF 4 LETTERS
BETWEEN A AND F

YOUR GUESS?	F F F F	2 BLK	0 WHT
YOUR GUESS?	F F E E	1 BLK	1 WHT
YOUR GUESS?	D F F D	1 BLK	1 WHT
YOUR GUESS?	C F C F	3 BLK	0 WHT
YOUR GUESS?	B F C F	3 BLK	0 WHT
YOUR GUESS?	A F C F	4 BLK	0 WHT

YOU WIN IN 6 TRIES!

PLAY AGAIN? N

*
 * SWITCH FOR 8080
 *
 *
 * PROGRAM COPYRIGHT (C) 1977 BY
 * TECHNICAL SYSTEMS CONSULTANTS
 * BOX 2574 W. LAFAYETTE, IN. 47906
 *
 * SWITCH IS A GAME OF LOGIC IN WHICH THE
 * COMPUTER GENERATES A SEQUENCE OF THE 9
 * DIGITS 1 THROUGH 9 AND YOUR TASK IS TO
 * REARRANGE THEM TO NUMERICAL ORDER IN THE
 * FEWEST POSSIBLE MOVES. REARRANGEMENT IS
 * DONE BY SPECIFYING THE NUMBER OF DIGITS
 * IN THE SEQUENCE TO BE REVERSED. IF,
 * FOR EXAMPLE, YOU TELL THE COMPUTER TO
 * SWITCH 6, THE SEQUENCE OF THE FIRST 6
 * DIGITS (FROM THE LEFT) WILL BE REVERSED.
 * WHEN THE SEQUENCE IS IN ORDER THE COMPUTER
 * TELLS YOU HOW MANY MOVES IT TOOK TO DO SO.
 *
 * SEE ATTACHED SAMPLE OUTPUT FOR AN EXAMPLE.
 *
 * THIS PROGRAM ASSUMES THAT THE USER HAS AN
 * I/O TERMINAL OF SOME TYPE AND ROUTINES TO
 * PRINT AN ASCII CHARACTER FROM THE A REGISTER
 * AND INPUT AN ASCII CHARACTER INTO THE A
 * REGISTER. THE USER MUST LOAD THE ADDRESS
 * OF THE OUTPUT ROUTINE AT LOCATION 1015 HEX
 * AND THE ADDRESS OF THE INPUT ROUTINE AT
 * 1018 HEX (IN NORMAL 8080 FORM). ALSO, THE
 * USER MUST INSTALL THE ENTRY ADDRESS OF HIS
 * MONITOR AT 101B HEX. THE SOURCE LISTING
 * IS SUPPLIED WITH THE ADDRESSES SET FOR THE
 * TSC 8080 MONITOR. THE STACK ADDRESS IS
 * SPECIFIED AT LOCATION 1001 HEX. IF IT IS
 * NECESSARY THIS ADDRESS CAN BE MODIFIED TO
 * SUIT YOUR PARTICULAR SYSTEM.
 *
 * THE STARTING ADDRESS OF THIS PROGRAM IS
 * 1000 HEX. WHEN THE COMPUTER ASKS TO PLAY
 * AGAIN, TYPE Y FOR YES OR N FOR NO.
 * CAUTION MUST BE EXERCISED THAT BYTES RNDM
 * AND RNDM+1 (LOCATIONS 1006 AND 1007 HEX)
 * ARE NOT BOTH ZERO. IF THIS IS THE CASE,
 * CHANGE ONE OR BOTH BYTES TO SOME NON-ZERO
 * VALUE BEFORE RUNNING THE PROGRAM.
 *
 *
 *

1000		ORG	1000H	
1000	31 EF 1F	LXI	S, 1FFFFH	SETUP STACK
1003	C3 53 10	JMP	BEGIN	START THE GAME
1006	01 02	RNDM	DS	2

1008 03/04/05/06/07/08/09/0A/0B STRNG DS 9
 1011 0C TRYS DS 1
 1012 3E 20 OUTS MVI A,20H PRINT A SPACE

* JUMP TO EXTERNAL ROUTINES

1014 C3 E2 00 OUTCH JMP 081F9H
 1017 C3 D5 00 INCH JMP 08205H
 101A C3 57 00 MONTR JMP 0801EH

* PRINT ROUTINES

101D CD 14 1020 PCHR CALL OUTCH PRINT CHARACTER
 1020 23 PDNXT INX H
 1021 7E PDATA MOV A,M GET CHARACTER
 1022 FE 04 CPI 4 CHECK FOR EOT
 1024 C2 1D 1020 JNZ PCHR
 1027 C9 RET
 1028 23 PSNXT INX H POINT TO NEXT STRING
 1029 CD 2F 1020 PSTRG CALL PCRLF PRINT CR AND LF
 102C C3 21 1020 JMP PDATA PRINT STRING
 102F E5 PCRLF PUSH H SAVE H AND L
 1030 21 54 1122 LXI H,CRLF
 1033 CD 21 1020 CALL PDATA PRINT CR AND LF
 1036 E1 POP H RESTORE H AND L
 1037 C9 RET

* RANDOM NUMBER GENERATOR

1038 E5 RANDM PUSH H SAVE H AND L
 1039 C5 PUSH B SAVE B REGISTER
 103A 21 06 1020 LXI H,RNDM POINT TO RNDM
 103D 06 08 MVI B,B SET LOOP COUNTER
 103F 7E RLOOP MOV A,M GET MS BYTE OF RNDM
 1040 07 RLC ALIGN BITS 13 AND 14
 1041 AE XRA M XOR WITH RNDM
 1042 17 RAL GET BIT 13.XOR.14
 1043 17 RAL INTO CARRY BIT
 1044 23 INX H
 1045 7E MOV A,M ROTATE BOTH
 1046 17 RAL BYTES OF RNDM
 1047 77 MOV M,A PUTTING CARRY
 1048 2B DCX H INTO LSB
 1049 7E MOV A,M
 104A 17 RAL
 104B 77 MOV M,A
 104C 05 DCR B
 104D C2 3F 1020 JNZ RLOOP DO LOOP 8 TIMES
 1050 C1 POP B RESTORE B REGISTER
 1051 E1 POP H RESTORE H AND L
 1052 C9 RET
 1053 CD 2F 1020 BEGIN CALL PCRLF
 1056 21 38 1122 LXI H,SWTCH
 1059 CD 29 1020 CALL PSTRG PRINT HEADER

105C	CD 2F 10 20	START	CALL	PCRLF	
105F	21 45 11 22		LXI	H, SEQ	
1062	CD 29 10 20		CALL	PSTRG	PRINT MESSAGE
1065	AF		XRA	A	
1066	32 11 10 20		STA	TRYS	TRY COUNTER SET TO 0
1069	4F		MOV	C, A	CLEAR REGISTERS
106A	57		MOV	D, A	
106B	59	LOOP1	MOV	E, C	
106C	CD 38 10 20	GETRN	CALL	RANDM	GET A RANDOM NUMBER
106F	E6 0F		ANI	0FH	
1071	CA 6C 10 20		JZ	GETRN	CHECK FOR = 0
1074	FE 0A		CPI	0AH	
1076	F2 6C 10 20		JP	GETRN	CHECK FOR > 9
1079	47		MOV	B, A	
107A	AF	LOOP2	XRA	A	
107B	BB		CMP	E	
107C	CA 8C 10 20		JZ	RNDOK	IF FIRST DIGIT, IT'S OK
107F	1D		DCR	E	
1080	21 08 10 20		LXI	H, STRNG	
1083	19		DAD	D	
1084	7E		MOV	A, M	
1085	B8		CMP	B	
1086	C2 7A 10 20		JNZ	LOOP2	CHECK FOR UNIQUE DIGIT
1089	C3 6B 10 20		JMP	LOOP1	IF NOT, GET ANOTHER
108C	59	RNDOK	MOV	E, C	
108D	21 08 10 20		LXI	H, STRNG	
1090	19		DAD	D	
1091	70		MOV	M, B	PUT DIGIT IN SEQUENCE
1092	79		MOV	A, C	
1093	FE 08		CPI	08	
1095	CA 9C 10 20		JZ	CONT	IF LAST DIGIT, CONTINUE
1098	0C		INR	C	
1099	C3 6B 10 20		JMP	LOOP1	ELSE, GET ANOTHER DIGIT
109C	CD 24 11 22	CONT	CALL	STROT	PRINT OUT STRING
109F	AF	PRMPT	XRA	A	
10A0	57		MOV	D, A	
10A1	21 5B 11 22		LXI	H, SWAMT	GET AMOUNT TO SWITCH
10A4	CD 21 10 20	INPUT	CALL	PDATA	
10A7	CD 17 10 20		CALL	INCH	
10AA	D6 30		SUI	30H	
10AC	FA B4 10 20		JM	ERR	
10AF	FE 0A		CPI	0AH	
10B1	FA BA 10 20		JM	OK	CHECK FOR VALID
10B4	21 66 11 22	ERR	LXI	H, QUES	
10B7	C3 A4 10 20		JMP	INPUT	
10BA	5F	OK	MOV	E, A	
10BB	1D		DCR	E	
10BC	21 08 10 20		LXI	H, STRNG	
10BF	19		DAD	D	
10C0	11 08 10 20		LXI	D, STRNG	
10C3	1F		RAR		NO. OF SWITCHES NEEDED
10C4	E6 07		ANI	07H	= AMOUNT/2. PUT IN A
10C6	CA D5 10 20		JZ	DONE	
10C9	46	DOSW	MOV	B, M	SWITCH OUTER DIGITS

10CA	EB		XCHG		
10CB	4E		MOV	C,M	
10CC	70		MOV	M,B	
10CD	EB		XCHG		
10CE	71		MOV	M,C	
10CF	13		INX	D	
10D0	2B		DCX	H	
10D1	3D		DCR	A	
10D2	C2 C9 1020		JNZ	DOSW	
10D5	CD 2F 1020	DONE	CALL	PCRLF	
10D8	CD 24 1122		CALL	STROT	PRINT OUT STRING
10DB	3A 11 1020		LDA	TRYS	
10DE	C6 01		ADI	01	
10E0	27		DAA		
10E1	32 11 1020		STA	TRYS	INC. TRY COUNTER
10E4	06 09		MVI	B,9	
10E6	21 10 1020		LXI	H,STRNG+8	
10E9	7E	CHKST	MOV	A,M	
10EA	B8		CMF	B	SEE IF DIGITS IN ORDER
10EB	C2 9F 1020		JNZ	PRMPT	IF NOT, DO NEXT SWITCH
10EE	2B		DCX	H	
10EF	05		DCR	B	
10F0	C2 E9 1020		JNZ	CHKST	
10F3	21 6A 1122		LXI	H,WIN	HE WINS!
10F6	CD 29 1020		CALL	PSTRG	OUTPUT IT

* ROUTINE TO PRINT NUMBER OF TRYS

10F9	3A 11 1020		LDA	TRYS	OUTPUT NO. OF TRYS
10FC	47		MOV	B,A	
10FD	E6 F0		ANI	0F0H	
10FF	CA 0E 1122		JZ	ONDGT	
1102	0F		RRC		
1103	0F		RRC		
1104	0F		RRC		
1105	0F		RRC		
1106	F6 30		ORI	30H	
1108	CD 14 1020		CALL	OUTCH	OUT TENS DIGIT OF TRYS
110B	78	ONDGT	MOV	A,B	
110C	E6 0F		ANI	0FH	
110E	F6 30		ORI	30H	
1110	CD 14 1020		CALL	OUTCH	OUT ONES DIGIT OF TRYS
1113	CD 20 1020		CALL	PDXNT	
1116	CD 28 1020		CALL	PSNXT	
1119	CD 17 1020		CALL	INCH	PLAY AGAIN?
111C	FE 4E		CPI	'N'	
111E	C2 5C 1020		JNZ	START	
1121	C3 1A 1020		JMP	MONTR	IF NOT, EXIT PROGRAM

* ROUTINE TO PRINT NUMBER STRING

1124	06 09	STROT	MVI	B,9	OUTPUT ENTIRE STRING
1126	21 08 20		LXI	H,STRNG	
1129	7E	NXTNO	MOV	A,M	

112A	C6 30	ADI	30H	
112C	CD 14 20	CALL	OUTCH	OUTPUT A DIGIT
112F	CD 12 20	CALL	OUTS	OUTPUT A SPACE
1132	23	INX	H	
1133	05	DCR	B	
1134	C2 29 22	JNZ	NXTNO	IF NOT DONE, GET NEXT ...
1137	C9	RET		

* PRINT STRINGS

57 49 54 43 48 20 2A 2A 223B

1138	2A 2A 20 53	SWTCH	DB	'** SWITCH **'
1144	04		DB	
1145	54 68 65 20	SEQ	DB	'THE SEQUENCE IS'
1154	0D 04 00 00	CRLF	DB	0DH, 0AH, 0, 0, 0, 0, 0
115B	20 20 53 77 69	SWAMT	DB	' SWITCH? '
1165	04		DB	4
1166	20 3F 20	QUES	DB	' ? '
1169	04		DB	4
116A	59 6F 55 20	WIN	DB	' YOU WIN IN '
1175	04		DB	4
1176	20 4D 6F 76 65 73		DB	' MOVES '
117C	04		DB	4
117D	50 6C 61 79 20 61 67 61 69 6E 3F 20		DB	' PLAY AGAIN? '
1189	04		DB	4
118A		END		

SYMBOLIC REFERENCE TABLE.

BEGIN	1053	PRMPT	109F
CHKST	10E9	PSNXT	1028
CONT	109C	PSTRG	1029
CRLF	1154	QUES	1166
DONE	10D5	RANDM	1038
DOSW	10C9	RLOOP	103F
ERR	10B4	RNDM	1006
GETRN	106C	RNDOK	108C
INCH	1017	SEQ	1145
INPUT	10A4	START	105C
LOOP1	106B	STRNG	1008
LOOP2	107A	STROT	1124
MONTR	101A	SWAMT	115B
NXTNO	1129	SWTCH	1138
OK	10BA	TRYS	1011
ONDGT	110B	WIN	116A
OUTCH	1014		
OUTS	1012		
PCHR	101D		
PCRLF	102F		
PDATA	1021		
PDNXT	1020		

SWITCH

THE SEQUENCE IS

8	7	4	2	5	9	6	3	1	SWITCH?	6
9	5	2	4	7	8	6	3	1	SWITCH?	9
1	3	6	8	7	4	2	5	9	SWITCH?	4
8	6	3	1	7	4	2	5	9	SWITCH?	8
5	2	4	7	1	3	6	8	9	SWITCH?	4
7	4	2	5	1	3	6	8	9	SWITCH?	7
6	3	1	5	2	4	7	8	9	SWITCH?	6
4	2	5	1	3	6	7	8	9	SWITCH?	3
5	2	4	1	3	6	7	8	9	SWITCH?	5
3	1	4	2	5	6	7	8	9	SWITCH?	3
4	1	3	2	5	6	7	8	9	SWITCH?	4
2	3	1	4	5	6	7	8	9	SWITCH?	2
3	2	1	4	5	6	7	8	9	SWITCH?	3
1	2	3	4	5	6	7	8	9		

SUCCESS IN 13 TRIES!

PLAY AGAIN? Y

SWITCH

THE SEQUENCE IS

5	9	7	1	4	2	3	6	8	SWITCH?	2
9	5	7	1	4	2	3	6	8	SWITCH?	9
8	6	3	2	4	1	7	5	9	SWITCH?	8
5	7	1	4	2	3	6	8	9	SWITCH?	2
7	5	1	4	2	3	6	8	9	SWITCH?	7
6	3	2	4	1	5	7	8	9	SWITCH?	6
5	1	4	2	3	6	7	8	9	SWITCH?	5
3	2	4	1	5	6	7	8	9	SWITCH?	3
4	2	3	1	5	6	7	8	9	SWITCH?	4
1	3	2	4	5	6	7	8	9	SWITCH?	2
3	1	2	4	5	6	7	8	9	SWITCH?	3
2	1	3	4	5	6	7	8	9	SWITCH?	2
1	2	3	4	5	6	7	8	9		

SUCCESS IN 12 TRIES!

PLAY AGAIN? Y

SWITCH

THE SEQUENCE IS

2	3	7	4	1	5	8	9	6	SWITCH?	8
9	8	5	1	4	7	3	2	6	SWITCH?	9
6	2	3	7	4	1	5	8	9	SWITCH?	4
7	3	2	6	4	1	5	8	9	SWITCH?	7
5	1	4	6	2	3	7	8	9	SWITCH?	4
6	4	1	5	2	3	7	8	9	SWITCH?	6
3	2	5	1	4	6	7	8	9	SWITCH?	3
5	2	3	1	4	6	7	8	9	SWITCH?	5
4	1	3	2	5	6	7	8	9	SWITCH?	4
2	3	1	4	5	6	7	8	9	SWITCH?	2
3	2	1	4	5	6	7	8	9	SWITCH?	3
1	2	3	4	5	6	7	8	9		

SUCCESS IN 11 TRIES!

PLAY AGAIN? Y

SWITCH

THE SEQUENCE IS

2	8	7	2	1	6	5	9	4	SWITCH?	8
9	5	6	1	2	7	8	3	4	SWITCH?	9
4	3	8	7	2	1	6	5	9	SWITCH?	3
8	3	4	7	2	1	6	5	9	SWITCH?	8
5	6	1	2	7	4	3	8	9	SWITCH?	5
7	2	1	6	5	4	3	8	9	SWITCH?	7
3	4	5	6	1	2	7	8	9	SWITCH?	4
6	5	4	3	1	2	7	8	9	SWITCH?	6
2	1	3	4	5	6	7	8	9	SWITCH?	2
1	2	3	4	5	6	7	8	9		

SUCCESS IN 9 TRIES!

PLAY AGAIN? N

KIM

2A2A 2A

*
* ACEY-DUCEY FOR 8080
*
*
* COPYRIGHT (C) 1977 BY
* TECHNICAL SYSTEMS CONSULTANTS
* BOX 2574 W. LAFAYETTE, IN. 47906
*
* ACEY-DUCEY IS A BETTING CARD GAME. YOU
* WILL BE GIVEN \$100 TO PLAY WITH. THE
* OBJECT IS TO BREAK THE BANK BY WINNING
* \$1000 OR MORE. IF YOU LOSE ALL YOUR MONEY,
* THE GAME WILL END.
*
* WHEN THE GAME IS PLAYED, TWO CARDS WILL BE
* DEALT FACE UP. YOU THEN PLACE A BET WHICH
* IS LESS THAN OR EQUAL TO THE AMOUNT OF
* MONEY YOU HAVE LEFT. TO DO SO, TYPE IN
* THE AMOUNT (IN WHOLE DOLLARS) AND THEN
* TYPE A CARRIAGE RETURN. YOU ARE BETTING
* THAT THE THIRD CARD YOU WILL RECEIVE WILL
* HAVE A VALUE BETWEEN YOUR FIRST TWO CARDS.
* IF IT IS, YOU WIN THE AMOUNT YOU BET. IF
* THE THIRD IS EQUAL TO ONE OF THE FIRST TWO
* OR OUTSIDE OF THE RANGE, YOU LOSE WHAT
* YOU BET. ACES ARE ALWAYS HIGH.
*
* SEE ATTACHED SAMPLE OUTPUT FOR AN EXAMPLE.
*
* THIS PROGRAM ASSUMES THAT THE USER HAS AN
* I/O TERMINAL OF SOME TYPE AND ROUTINES TO
* PRINT AN ASCII CHARACTER FROM THE A REGISTER
* AND INPUT AN ASCII CHARACTER INTO THE A
* REGISTER. THE USER MUST LOAD THE ADDRESS
* OF THE OUTPUT ROUTINE AT LOCATION 1012 HEX
* AND THE ADDRESS OF THE INPUT ROUTINE AT
* 1015 HEX (IN NORMAL 8080 FORM). ALSO, THE
* USER MUST INSTALL THE ENTRY ADDRESS OF HIS
* MONITOR AT 1018 HEX. THE SOURCE LISTING
* IS SUPPLIED WITH THE ADDRESSES SET FOR THE
* TSC 8080 MONITOR. THE STACK ADDRESS IS
* SPECIFIED AT LOCATION 1001 HEX. IF IT IS
* NECESSARY THIS ADDRESS CAN BE MODIFIED TO
* SUIT YOUR PARTICULAR SYSTEM.
*
* THE STARTING ADDRESS OF THIS PROGRAM IS
* 1000 HEX. WHEN THE COMPUTER ASKS TO PLAY
* AGAIN, TYPE Y FOR YES OR N FOR NO.
* CAUTION MUST BE EXERCISED THAT BYTES RNDM
* AND RNDM+1 (LOCATIONS 1006 AND 1007 HEX)
* ARE NOT BOTH ZERO. IF THIS IS THE CASE,
* CHANGE ONE OR BOTH BYTES TO SOME NON-ZERO
* VALUE BEFORE RUNNING THE PROGRAM.

*
 * GOOD LUCK !!!
 *
 *

1000		ORG	1000H	
1000	31 FF 1F	LXI	S,1FFFFH	SETUP STACK
1003	C3 50 20	JMP	BEGIN	START THE GAME
1006		RNDM DS	2	
1008		MONEY DS	2	
100A		CARD1 DS	1	
100B		CARD2 DS	1	
100C	54 4A 51 4B 41	TABLE DB	'TJQKA'	

* JUMP TO EXTERNAL ROUTINES

1011	^{24 0C} C3 F9 81	OUTCH JMP	081F9H
1014	C3 05 82 ^{1E 23}	INCH JMP	08205H
1017	C3 1E 80 ^{0 0}	MONTR JMP	0801EH

* PRINT ROUTINES

101A	CD 11 20	PCHR CALL	OUTCH	PRINT CHARACTER
101D	23	PDXNT INX	H	
101E	7E	PDATA MOV	A,M	GET CHARACTER
101F	FE 04	CPI	4	CHECK FOR EOT
1021	C2 1A 20	JNZ	PCHR	
1024	C9	RET		
1025	23	PSNXT INX	H	POINT TO NEXT STRING
1026	CD 2C 20	PSTRG CALL	PCRLF	PRINT CR AND LF
1029	C3 1E 20	JMP	PDATA	PRINT STRING
102C	E5	PCRLF PUSH	H	SAVE H AND L
102D	21 18 22	LXI	H,CRLF	
1030	CD 1E 20	CALL	PDATA	PRINT CR AND LF
1033	E1	POP	H	RESTORE H AND L
1034	C9	RET		

* RANDOM NUMBER GENERATOR

1035	E5	RANDM PUSH	H	SAVE H AND L
1036	C5	PUSH	B	SAVE B REGISTER
1037	21 06 20	LXI	H,RNDM	POINT TO RNDM
103A	06 08	MVI	B,8	SET LOOP COUNTER
103C	7E	RLOOP MOV	A,M	GET MS BYTE OF RNDM
103D	07	RLC		ALIGN BITS 13 AND 14
103E	AE	XRA	M	XOR WITH RNDM
103F	17	RAL		GET BIT 13.XOR.14
1040	17	RAL		INTO CARRY BIT
1041	23	INX	H	
1042	7E	MOV	A,M	ROTATE BOTH
1043	17	RAL		BYTES OF RNDM
1044	77	MOV	M,A	PUTTING CARRY
1045	2B	DCX	H	INTO LSB
1046	7E	MOV	A,M	
1047	17	RAL		

1048	77		MOV	M,A	
1049	05		DCR	B	
104A	C2 3C 20		JNZ	RLOOP	DO LOOP 8 TIMES
104D	C1		POP	B	RESTORE B REGISTER
104E	E1		POP	H	RESTORE H AND L
104F	C9		RET		
1050	0E 03	BEGIN	MVI	C,3	INIT. SCARED COUNT
1052	CD 2C 20		CALL	PCRLF	
1055	21 1F 22		LXI	H,INTRO	
1058	CD 26 20		CALL	PSTRG	PRINT INTRO
105B	3E 01		MVI	A,1	SETUP INITIAL MONEY
105D	32 08 20		STA	MONEY	TO 100 DOLLARS
1060	AF		XRA	A	
1061	32 09 20		STA	MONEY+1	
1064	CD 2C 20	START	CALL	PCRLF	
1067	21 E9 21		LXI	H,MONYS	OUTPUT AMOUNT OF MONEY
106A	CD 26 20		CALL	PSTRG	
106D	CD 9B 21		CALL	OUTMN	
1070	21 F3 21		LXI	H,MONS1	
1073	CD 1E 20		CALL	PDATA	
1076	3A 08 20		LDA	MONEY	SEE IF HE BROKE BANK
1079	FE 10		CPI	10H	
107B	FA 8D 20		JM	STRT1	IF NOT, CONTINUE
107E	21 E9 22		LXI	H,BROKB	ELSE, REPORT IT
1081	CD 26 20		CALL	PSTRG	
1084	CD 25 20		CALL	PSNXT	
1087	21 B8 22		LXI	H,PLYAG	PLAY AGAIN?
108A	C3 40 21		JMP	NEWGM	
108D	21 FD 21	STRT1	LXI	H,MONS2	
1090	CD 26 20		CALL	PSTRG	ANNOUNCE CARDS
1093	CD 51 21		CALL	GETCD	GET FIRST CARD
1096	32 0A 20		STA	CARD1	SAVE IT AT CARD1
1099	CD 2C 20		CALL	PCRLF	
109C	CD 51 21		CALL	GETCD	GET SECOND CARD
109F	32 0B 20		STA	CARD2	SAVE IT AT CARD2
10A2	21 33 22	AGAIN	LXI	H,GETBT	
10A5	CD 26 20		CALL	PSTRG	OUTPUT BET PROMPT
10A8	CD 74 21		CALL	INBET	GET THE BET
10AB	AF		XRA	A	
10AC	82		ADD	D	
10AD	C2 D9 20		JNZ	CONT	
10B0	83		ADD	E	
10B1	C2 D9 20		JNZ	CONT	IF BET IS NONZERO, CONT
10B4	2A 0A 20		LHLD	CARD1	ELSE, SEE IF CHICKEN
10B7	7D		MOV	A,L	
10B8	94		SUB	H	TAKE CARD1 - CARD2
10B9	FE 04		CPI	4	
10BB	F2 C3 20		JP	SCARD	
10BE	FE FD		CPI	OFDH	IF DIFFERENCE IS 3
10C0	F2 64 20		JP	START	OR LESS, GO BACK
10C3	0D	SCARD	DCR	C	DEC. SCARED COUNT
10C4	FA D0 20		JM	SCRD2	TOO MANY TIMES?
10C7	21 46 22		LXI	H,CHICK	IF NOT, PRINT CHICKEN
10CA	CD 26 20		CALL	PSTRG	

10CD	C3 64 20		JMP	START	GO REPEAT
10D0	21 51 22	SCRD2	LXI	H,CHIC2	PRINT TOO CHICKEN!
10D3	CD 26 20		CALL	PSTRG	
10D6	C3 A2 20		JMP	AGAIN	FORCE A BET
10D9	3A 08 20	CONT	LDA	MONEY	
10DC	BA		CMP	D	DID HE BET TOO MUCH?
10DD	DA 07 21		JC	TOMCH	IF SO REPORT IT
10E0	C2 EA 20		JNZ	CONT1	
10E3	3A 09 20		LDA	MONEY+1	
10E6	BB		CMP	E	
10E7	DA 07 21		JC	TOMCH	
10EA	21 70 22	CONT1	LXI	H,NXTCD	ANNOUNCE THE 3RD CARD
10ED	CD 26 20		CALL	PSTRG	
10F0	CD 51 21		CALL	GETCD	GET CARD 3
10F3	2A 0A 20		LHLD	CARD1	
10F6	BD		CMP	L	COMPARE CARD3 AND CARD1
10F7	CA 21 21		JZ	LOSE	IF C3 = C1, HE LOSES
10FA	F2 10 21		JP	TST2	IF C3>C1, GO TO TST2
10FD	BC		CMP	H	ELSE, COMPARE C3 AND C2
10FE	CA 21 21		JZ	LOSE	IF C3 = C2, HE LOSES
1101	FA 21 21		JM	LOSE	IF C3<C2, HE LOSES
1104	C3 14 21		JMP	WIN	ELSE, HE WINS!
1107	21 CB 22	TOMCH	LXI	H,TMUCH	HE BET TOO MUCH
110A	CD 26 20		CALL	PSTRG	REPORT IT
110D	C3 A2 20		JMP	AGAIN	
1110	BC	TST2	CMP	H	COMPARE C3 AND C2
1111	F2 21 21		JP	LOSE	IF C3<C2, HE LOSES

* WIN ROUTINE

1114	21 83 22	WIN	LXI	H,WINST	HE WINS!
1117	CD 26 20		CALL	PSTRG	OUTPUT IT
111A	AF		XRA	A	CLEAR CARRY FOR ADDING
111B	CD D9 21		CALL	FIXBT	ADD BET TO MONEY
111E	C3 64 20		JMP	START	GO REPEAT

* LOSE ROUTINE

1121	21 8F 22	LOSE	LXI	H,LOSST	HE LOST!
1124	CD 26 20		CALL	PSTRG	OUTPUT IT
1127	CD D0 21		CALL	SUBBT	SUBTRACT BET FROM MONEY
112A	2A 08 20		LHLD	MONEY	SEE IF MONEY'S GONE
112D	AF		XRA	A	
112E	85		ADD	L	
112F	C2 64 20		JNZ	START	
1132	84		ADD	H	
1133	C2 64 20		JNZ	START	IF NOT, GO REPEAT
1136	CD 2C 20		CALL	PCRLF	
1139	21 A0 22		LXI	H,NOMON	ELSE, REPORT NO MONEY
113C	CD 26 20		CALL	PSTRG	
113F	23		INX	H	
1140	CD 2C 20	NEWGM	CALL	PCRLF	
1143	CD 26 20		CALL	PSTRG	
1146	CD 14 20		CALL	INCH	PLAY AGAIN?

1149	FE 4E	CPI	'N'	
114B	C2 50 10	JNZ	BEGIN	IF SO, START OVER
114E	C3 17 10	JMP	MONTR	IF NOT, EXIT PROGRAM

* GET A CARD ROUTINE (AND OUTPUT IT)

1151	CD 35 10	GETCD	CALL	RANDM	GET A RANDOM NO.
1154	E6 0F		ANI	0FH	MASK IT OFF
1156	FE 0D		CPI	0DH	IS IT LESS THAN 13?
1158	F2 51 11		JP	GETCD	IF NOT, GET ANOTHER
115B	47		MOV	B,A	SAVE NUMERICAL VALUE
115C	C6 32		ADI	32H	MAKE IT ASCII
115E	FE 3A		CPI	3AH	IF CARD IS 2 THRU 9,
1160	FA 6F 11		JM	OTCRD	OUTPUT IT
1163	D6 3A		SUI	3AH	
1165	21 0C 10		LXI	H, TABLE	POINT TO TABLE,
1168	85		ADD	L	THEN MOVE TO PROPER
1169	6F		MOV	L,A	PLACE IN THE TABLE
116A	D2 6E 11		JNC	CARD	
116D	24		INR	H	
116E	7E	CARD	MOV	A,M	PUT PROPER CHAR. IN A
116F	CD 11 10	OTCRD	CALL	OUTCH	OUTPUT CARD
1172	78		MOV	A,B	RESTORE NUMERICAL VALUE
1173	C9		RET		

* INPUT BET ROUTINE

1174	11 00 00	INBET	LXI	D,0	CLEAR D AND E
1177	CD 14 10	INBT1	CALL	INCH	GET A CHARACTER
117A	FE 0D		CPI	0DH	IS IT A CR
117C	C8		RZ	IF SO, RETURN	
117D	D6 30		SUI	30H	IS IT VALID?
117F	FA 92 11		JM	ERROR	
1182	FE 0A		CPI	10	
1184	F2 92 11		JP	ERROR	IF NOT, REPORT ERROR
1187	EB		XCHG		ROTATE D/E REGISTER
1188	29		DAD	H	LEFT 4 PLACES
1189	29		DAD	H	
118A	29		DAD	H	
118B	29		DAD	H	
118C	EB		XCHG		
118D	83		ADD	E	ADD IN L.S. DIGIT
118E	5F		MOV	E,A	
118F	C3 77 11		JMP	INBT1	GO GET ANOTHER CHAR.
1192	21 E5 12	ERROR	LXI	H,ERR	
1195	CD 1E 10		CALL	PDATA	OUTPUT ERROR
1198	C3 74 11		JMP	INBET	

* OUTPUT MONEY ROUTINE

119B	06 00	OUTMN	MVI	B,0	CLEAR B
119D	2A 08 10		LHLD	MONEY	
11A0	AF		XRA	A	
11A1	85		ADD	L	CHECK MONEY

11A2	CA B2 21	JZ	OTBT2	IF ZERO, CONTINUE
11A5	E6 F0	ANI	OFOH	MASK TOP HALF
11A7	CA AD 21	JZ	OTBT1	IF ZERO, CONTINUE
11AA	CD C5 21	CALL	OUTH	OUT M.S. HALF OF MONEY
11AD	7D	OTBT1	MOV A,L	
11AE	CD C9 21	CALL	OUTHR	OUT L.S. HALF OF MONEY
11B1	04	INR	B	
11B2	AF	OTBT2	XRA A	
11B3	80	ADD	B	HAVE WE OUTPUT A CHAR?
11B4	7C	MOV	A,H	
11B5	C2 BD 21	JNZ	NOZRO	
11B8	E6 F0	ANI	OFOH	
11BA	CA C0 21	JZ	OTBT3	CHECK FOR ZERO
11BD	CD C5 21	NOZRO	CALL	OUT MS HALF OF MONEY+1
11C0	7C	OTBT3	MOV A,H	
11C1	CD C9 21	CALL	OUTHR	OUT LS HALF OF MONEY+1
11C4	C9	RET		
11C5	1F	OUTH	RAR	SHIFT DIGIT TO
11C6	1F	RAR		RIGHT HALF
11C7	1F	RAR		
11C8	1F	RAR		
11C9	E6 OF	OUTHR	ANI OFH	
11CB	C6 30	ADI	30H	MAKE IT ASCII
11CD	C3 11 20	JMP	OUTCH	OUTPUT CHARACTER

* SUBTRACT AND ADD BET ROUTINES

11D0	3E 99	SUBBT	MVI A,99H	TENS COMP. THE BET
11D2	92	SUB	D	
11D3	57	MOV	D,A	
11D4	3E 99	MVI	A,99H	
11D6	93	SUB	E	
11D7	5F	MOV	E,A	
11D8	37	STC		SET CARRY FOR 2'S COMP.
11D9	2A 08 10	FIXBT	LHLD MONEY	GET MONEY IN H AND L
11DC	7C	MOV	A,H	AND ADD BET TO IT
11DD	8B	ADC	E	
11DE	27	DAA		
11DF	32 09 20	STA	MONEY+1	
11E2	7D	MOV	A,L	
11E3	8A	ADC	D	
11E4	27	DAA		
11E5	32 08 20	STA	MONEY	PUT RESULT BACK
11E8	C9	RET		

* PRINT STRINGS

11E9	59 4F 55 20	MONYS	DB	'YOU HAVE '
11F2	04		DB	4
11F3	20 44 4F 4C	MONS1	DB	'20 DOLLARS.'
11FC	04		DB	4
11FD	48 45 52 45	MONS2	DB	'HERE ARE YOUR NEXT 2 CARDS!'
1218	0D,0A,00,00	CRLF	DB	0D,0A,0,0,0,0,4
121E	0C 41 43 45 59	INTRO	DB	'ACEY DUCEY FOR 8080'

1232	04		DB	4	62 74 69 20 65 72 62 74 20
1233	57 48 41 54	GETBT	DB		'WHAT IS YOUR BET?'
1245	04		DB	4	
1246	43 48 49 43	CHICK	DB		'CHICKEN!!!'
1250	04		DB	4	
1251	59 4F 55	CHIC2	DB		'YOU'
1254	27		DB	27H	
1255	56 45 20 43		DB		'VE CHICKENED OUT TOO MUCH!'
126F	04		DB	4	
1270	59 4F 55 52	NXTCD	DB		'YOUR NEXT CARD IS'
1282	04		DB	4	
1283	59 4F 55 20	WINST	DB		'YOU WIN !!!'
128E	04		DB	4	
128F	53 4F 52 52	LOSST	DB		'SORRY, YOU LOSE.'
129F	04		DB	4	
12A0	59 4F 55 20	NOMON	DB		'YOU JUST BLEW YOUR WAD!'
12B7	04		DB	4	
12B8	50 4C 41 59	PLYAG	DB		'PLAY AGAIN (Y-N)?'
12CA	04		DB	4	
12CB	59 4F 55 20	TMUCH	DB		'YOU DON'
12D2	27		DB	27H	
12D3	54 20 48 41		DB		'T HAVE THAT MUCH!'
12E4	04		DB	4	
12E5	20 3F 20	ERR	DB		' ? '
12E8	04		DB	4	
12E9	2A 2A 2A 20	BROKB	DB		'*** CONGRATULATIONS ***'
1300	07 07 04		DB	7,7,4	
1303	59 4F 55 20		DB		'YOU JUST BROKE THE BANK!!!'
131C	07 04		DB	7,4	
131E	CD 20 0C C3 24 0C	INCHECHO	END		

2324 on a free ATM

SYMBOLIC REFERENCE TABLE.

AGAIN	10A2	MONS1	11F3	RANDM	1035
BEGIN	1050	MONS2	11FD	RLOOP	103C
BROKB	12E9	MONTR	1017	RNDM	1006
CARD	116E	MONYS	11E9	SCARD	10C3
CARD1	100A	NEWGM	1140	SCRD2	10D0
CARD2	100B	NOMON	12A0	START	1064
CHICK	1246	NOZRO	11BD	STRT1	108D
CHIC2	1251	NXTCD	1270	SUBBT	11D0
CONT	10D9	OTBT1	11AD	TABLE	100C
CONT1	10EA	OTBT2	11B2	TMUCH	12CB
CRLF	1218	OTBT3	11C0	TOMCH	1107
ERR	12E5	OTCRD	116F	TST2	1110
ERROR	1192	OUTCH	1011	WIN	1114
FIXBT	11D9	OUTHL	11C5	WINST	1283
GETBT	1233	OUTHR	11C9		
GETCD	1151	OUTMN	119B		
INBET	1174	PCHR	101A		
INBT1	1177	PCRLF	102C		
INCH	1014	PDATA	101E		
INTRO	121F	PDNXT	101D		
LOSE	1121	PLYAG	12B8		
LOSST	128F	PSNXT	1025		
MONEY	100B	PSTRG	1026		

:0610000031FF1FC3501078
:10100C00544A514B41C3F981C30582C31E80CD1193
:10101C0010237EFE04C21A10C923CD2C10C31E103F
:10102C00E5211812CD1E10E1C9E5C52106100608F0
:10103C007E07AE1717237E17772B7E177705C23CDA
:10104C0010C1E1C90E03CD2C10211F12CD26103E6C
:10105C0001320810AF320910CD2C1021E911CD2628
:10106C0010CD9B1121F311CD1E103A0810FE10FA71
:10107C008D1021E912CD2610CD251021B812C340B8
:10108C001121FD11CD2610CD5111320A10CD2C108D
:10109C00CD5111320B10213312CD2610CD7411AF5E
:1010AC0082C2D91083C2D9102A0A107D94FE04F290
:1010BC00C310FEFDF264100DFAD010214612CD269D
:1010CC0010C36410215112CD2610C3A2103A08107F
:1010DC00BADA0711C2EA103A0910BBD0A071121700B
:1010EC0012CD2610CD51112A0A10BDCA2111F210B1
:1010FC0011BCCA2111FA2111C3141121CB12CD2616
:10110C0010C3A210BCF22111218312CD2610AFCD39
:10111C00D911C36410218F12CD2610CDD0112A08FD
:10112C0010AF85C2641084C26410CD2C1021A012A3
:10113C00CD261023CD2C10CD2610CD1410FE4EC272
:10114C005010C31710CD3510E60FFE0DF25111479C
:10115C00C632FE3AFA6F11D63A210C10856FD26E58
:10116C0011247ECD111078C9110000CD1410FE0D84
:10117C00C8D630FA9211FE0AF29211EB29292929CC
:10118C00EB835FC3771121E512CD1E10C3741106DA
:10119C00002A0810AF85CAB211E6F0CAAD11CDC550
:1011AC00117DCDC91104AF807CC2BD11E6F0CAC05F
:1011BC0011CDC5117CCDC911C91F1F1F1FE60FC64C
:1011CC0030C311103E9992573E99935F372A0810FD
:1011DC007C8B273209107D8A27320810C9594F554C
:1011EC002048415645200420444F4C4C4152532E2C
:1011FC0004484552452041524520594F5552204EE6
:10120C004558542032204341524453210D0A0000CA
:10121C000000044143455920445543455920464F4D
:10122C005220383038300457484154204953205903
:10123C004F5552204245543F2004434849434B45A7
:10124C004E21212104594F5552756452043484943E7
:10125C004B454E4544204F555420544F4F204D552F
:10126C0043482104594F5552204E45585420434170
:10127C0052442049532004594F552057494E2021A0
:10128C00212104534F5252592C20594F55204C4F69
:10129C0053452E04594F55204A55535420424C4522
:1012AC005720594F5552205741442104504C415915
:1012BC0020414741494E2028592D4E293F200459A1
:1012CC004F5520444F4E27542048415645205448F2
:1012DC004154204D5543482104203F20042A2A2AFA
:1012EC0020434F4E47524154554C4154494F4E5355
:1012FC00202A2A2A070704594F55204A55535420AF
:10130C0042524F4B45205448452042414E4B2121DF
:02131C000704C4
**
**
:0000000000
#EOR
#EOF

ACEY-DUCEY
YOU NOW HAVE \$100

YOUR FIRST 2 CARDS ARE: 9 8
WHAT IS YOUR BET?

YOUR FIRST 2 CARDS ARE: J 7
WHAT IS YOUR BET? 5
YOUR THIRD CARD IS: 4
SORRY, YOU LOSE.
YOU NOW HAVE \$95

YOUR FIRST 2 CARDS ARE: J 7
WHAT IS YOUR BET?

YOUR FIRST 2 CARDS ARE: A K
WHAT IS YOUR BET?

YOUR FIRST 2 CARDS ARE: J 6
WHAT IS YOUR BET? 50
YOUR THIRD CARD IS: 8
YOU WIN!
YOU NOW HAVE \$145

YOUR FIRST 2 CARDS ARE: 9 3
WHAT IS YOUR BET? 500
YOU DON'T HAVE THAT MUCH!
YOU NOW HAVE \$145
WHAT IS YOUR BET? 50
YOUR THIRD CARD IS: 6
YOU WIN!
YOU NOW HAVE \$195

YOUR FIRST 2 CARDS ARE: K 3
WHAT IS YOUR BET? 190
YOUR THIRD CARD IS: T
YOU WIN!
YOU NOW HAVE \$385

YOUR FIRST 2 CARDS ARE: 4 2
WHAT IS YOUR BET?

YOUR FIRST 2 CARDS ARE: A J
WHAT IS YOUR BET?

YOUR FIRST 2 CARDS ARE: 5 A
WHAT IS YOUR BET? 385
YOUR THIRD CARD IS: 0
YOU WIN!
YOU NOW HAVE \$770

YOUR FIRST 2 CARDS ARE: 7 4
WHAT IS YOUR BET? 770
YOUR THIRD CARD IS: 3
SORRY, YOU LOSE.
YOU NOW HAVE \$0
YOUR MONEY IS ALL GONE!
PLAY AGAIN? N

7. Software

7.1 Video Typewriter:

Both the input to and the output from a computer is ordinarily a string of characters, whether it be characters typed in from a typewriter-like keyboard or output from the computer to a printer. Not all of these "characters," however, strictly correspond to a printed symbol, like a letter. Consider the output to a printer. Some "characters" will cause the printer to perform some function other than a keystroke -- such as carriage return or backspace.

The VTI is essentially a block of memory, and at the hardware level does not distinguish between characters and other functions. Without an intervening program, the VTI would send a "carriage return" on to the screen or a symbol, rather than returning to the beginning of the line.

We include here a program that accepts a string of ASCII characters and causes them to appear on the screen exactly as the characters would be printed by a printer. "Carriage return" causes the cursor to return to the beginning of the line, "line feed" causes it to move down one line, and so forth.

The program includes a keyboard input routine, which puts the characters you type on the keyboard directly onto the screen, with proper carriage return, line feed, and other functions. Load the program as written. To use the computer as a "TV typewriter," connect the keyboard to the parallel input port provided on the video board.

This program when executed at address 0000 causes characters typed in at the keyboard to appear on the screen as they would be printed by a printer.*

The principal usefulness of the program is to interpret the output of another program which would ordinarily be sent on to a printer, so as to put the appropriate visual display on the screen.

*This program assumes the user has a defined stack area. If you have no preassigned stack location, execute a LXI SP, 0FFFH.

Programs ordinarily send a character from the accumulator to a serial output port in response to the instruction "out". The following program includes a subroutine called "out," located at address 1D00H. When called, this subroutine interprets the character in the accumulator as required to put it on the screen. In converting a program to run with the VTI, substitute "call out" for the output instruction.

VIDEO TERMINAL SOFTWARE - COMMAND SUMMARY

	Control Character	Function
Cursor Controls	H	Home Cursor
	R	Cursor Right
	L	Cursor Left
	U	Cursor Up
	D	Cursor Down
	E	Erase Screen
	X	Delete character
Mode Commands	I	Insert/delete mode set
	T	Text (reset I/D mode)
	F	auto line Feed mode set
	N	Normal TTY (reset ALF mode)
	S	Scroll mode set
	P	Page (reset scroll mode)

Line feed advances cursor one line, exception last line in scroll mode; then cursor fixed, and page scrolls.

Carriage return retreats cursor to beginning of line, blanking line from end unless I/D mode set.

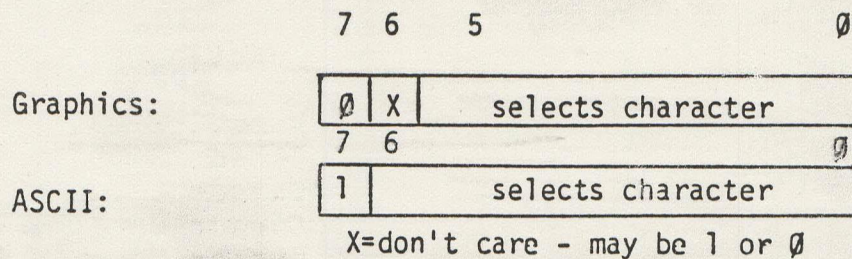
7.2 Graphics

The PolyMorphic VTI includes full graphics capability. Any or all character locations on the screen can be used in a graphics display.

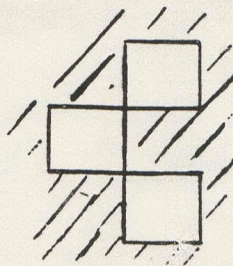
When a screen location is part of a graphics display, it is subdivided into six parts, thus:

5	2
4	1
3	0

(NOTE: Graphics display uses the entire screen location, including the border area that is kept dark to provide space around other characters). Each of the six "cells" of the screen location corresponds to one bit in the byte stored in the screen location. The "zero bit" corresponds to cell 0, etc.:



0 is "on" or "bright," 1 "off" or "dark." Thus, storing 01101010B (6AH) at a screen location produced this graphic at that location:



In the appendix is a chart of all 64 possible graphics characters, with their associated hex values.

The following "game" program, called LIFE, originally invented by John Conway and popularized by Martin Gardiner in his "Mathematical Games" Section of Scientific American in 1970, illustrates the power of the graphics capability.

LIFE depicts the birth, growth, and death of a culture of cells. When a cell has one neighbor or no neighbors in the eight cells adjacent to it, it dies of loneliness. When it has four or more neighbors in the eight adjacent cells, it dies of overcrowding. It survives into the next generation whenever it has two or three neighbors. So a cell may live for just one generation, or may live for as long as the culture lives (or anything in between). A cell is born whenever an empty cell location has exactly three neighbors. (Cells are trisexual.)

The game begins with an initial entry, or Divine Creation, of a seed organism (group of cells). The initial entry can be as simple or complex as you like. The life cycle of the resulting culture arises entirely from the nature of the initial entry given the rules of LIFE.

The following program executes the rules of LIFE on the video screen in graphics. Load both programs at the addresses indicated. Execute the screen clearing routine at 0F00. If your system has a stack area already allocated, then you need not set the stack pointer. If the stack is not already initialized, set it with a LXI SP, 0FFFH. Then you are ready to load an initial generation (by using the hex-to-equal-graphic table in appendix D) memory locations in the middle of the screen (such as 8A10H). When you are satisfied with your initial organism, execute the LIFE routine at address zero.

Video Typewriter Routine

Hexidecimal Address	Op Code	Mnemonic Instruction	Comments
0000		0100 SCRN EQU 8800H	*VIDEO SCREEN ADDRESS
0000		0110 STR EQU 1CFFH	*STORAGE FOR SYMBOL UNDER CURS
0000		0120 STS EQU 1CFEH	*STORE OUTPUT MODE
0000		0130 CURS EQU 1CFCH	*STORE RELATIVE CURSOR LOCAT
0000		0140 SEND EQU 8CH	*1ST BYTE OF SCREEN END
0000		0150 LINE EQU 64	*LINE LENGTH
0000		0160 CS EQU 0FFH	*CURSOR SYMBOL (RUB OUT)
0000		0170 LT EQU 3FH	*LINE TERMINATION CHARACTER
0000		0180 KBD EQU 88H	*KEYBOARD PORT ON VTI
0000		0190 ORG 0000	
0000 21 00 00		0200 LXI H, 0	
0003 22 FC 1C		0210 SHLD CURS	
0006 7D		0220 MOV A, L	
0007 32 FE 1C		0230 STA STS	
000A 21 11 00		0240 LXI H, LOOP	*SET UP WITH CLEAR SCREEN
000D E5		0250 PUSH H	*AND CURSOR AT UPPER RIGHT
000E C3 65 1D		0260 JMP FF	*USER MUST DEFINE OWN STACK
0011 FB		0310 LOOP EI	
0012 C3 11 00		0320 JMP LOOP	
0015		0330 ORG 38H	
0038 DB 88 (700, 1000)		0340 IN IN KBD	*RESTART 7
003A F6 80		0345 ORI 80H	*INTERRUPT DRIVEN KEYBOARD
003C F6 80		0350 ORI 88H	
003E 47		0360 MOV B, A	
003F CD 00 1D		0370 CALL OUT	
0042 78		0380 MOV A, B	
0043 C9		0400 RET	
0044		0500 ORG 1D00H	
1D00 2A FC 1C		1000 OUT LHLD CURS	
1D03 EB		1010 XCHG	
1D04 21 00 88		1020 LXI H, SCRN	*PUT RELATIVE CURSOR IN D
1D07 19		1030 DAD D	*PUT SCREEN BLOCK ADDRESS IN
1D08 47		1040 MOV B, A	*GET ABS CURSOR LOCATION
1D09 3A FF 1C		1050 LDA STR	
1D0C 77		1060 MOV M, A	
1D0D 78		1070 MOV A, B	*PUT BACK CHAR UNDER CURSOR
1D0E FE 88		1100 CPI 88H	*CHECK*
1D10 CA 5C 1D		1110 JZ HOME	*CTL H FOR HOME
1D13 FE 85		1120 CPI 85H	
1D15 CA 65 1D		1130 JZ FF	*CTL E FOR ERASE
1D18 FE 92		1140 CPI 92H	
1D1A CA 74 1D		1150 JZ HT	*CTL R FOR RIGHT
1D1D FE 95		1160 CPI 95H	
1D1F CA 7C 1D		1170 JZ VT	*CTL U FOR UP
1D22 FE 8C		1180 CPI 8CH	
1D24 CA 91 1D		1190 JZ BS	*CTL L FOR LEFT
1D27 FE 84		1192 CPI 84H	
1D29 CA E8 1D		1194 JZ LF	*CTL D FOR DOWN
1D2C FE 98		1200 CPI 98H	
1D2E CA 99 1D		1210 JZ RO	*CTL X (DELETE CHAR)

1D31 FE 89
 1D33 CA 86 1D
 1D36 FE 94
 1D38 CA 81 1D
 1D3B FE 86
 1D3D CA BC 1D
 1D40 FE 8E
 1D42 CA C7 1D
 1D45 FE 93
 1D47 CA D2 1D
 1D4A FE 90
 1D4C CA DD 1D
 1D4F FE 8A
 1D51 CA E8 1D
 1D54 FE 8D
 1D56 CA 21 1E
 1D59 C3 45 1E
 1D5C 21 00 00
 1D5F 22 FC 1C
 1D62 C3 6F 1E
 1D65 21 00 88
 1D68 36 3F
 1D6A 23
 1D6B 7C
 1D6C FE 8C
 1D6E C2 68 1D
 1D71 C3 5C 1D
 1D74 13
 1D75 EB
 1D76 22 FC 1C
 1D79 C3 6F 1E
 1D7C 21 C0 FF
 1D7F 19
 1D80 22 FC 1C
 1D83 C3 6F 1E
 1D86 3A FE 1C
 1D89 F6 01
 1D8B 32 FE 1C
 1D8E C3 6F 1E
 1D91 1B
 1D92 EB
 1D93 22 FC 1C
 1D96 C3 6F 1E
 1D99 3A FE 1C
 1D9C 1F
 1D9D D2 91 1D
 1DA0 23
 1DA1 7E
 1DA2 2B
 1DA3 77
 1DA4 23
 1DA5 7D
 1DA6 E6 3F

1220 CPI 89H
 1230 JZ SID
 1240 CPI 94H
 1250 JZ RID
 1260 CPI 86H
 1270 JZ SALF
 1271 CPI 8EH
 1272 JZ RALF
 1280 CPI 93H
 1290 JZ SSC
 1300 CPI 90H
 1310 JZ RSC
 1320 CPI 8AH
 1330 JZ LF
 1340 CPI 8DH
 1350 JZ CR
 1360 JMP DEF
 2000 HOME LXI H, 0
 2010 SHLD CURS
 2020 JMP OUT1
 2030 FF LXI H, SCRN
 2050 WIPE MVI M, LT
 2060 INX H
 2070 MOV A, H
 2080 CPI SEND
 2090 JNZ WIPE
 2100 JMP HOME
 2110 HT INX D
 2120 XCHG
 2130 SHLD CURS
 2140 JMP OUT1
 2150 VT LXI H, 0-LINE
 2160 DAD D
 2170 SHLD CURS
 2180 JMP OUT1
 2190 SID LDA STS
 2200 ORI 01H
 2210 STA STS
 2220 JMP OUT1
 2230 BS DCX D
 2240 XCHG
 2250 SHLD CURS
 2260 JMP OUT1
 2270 RO LDA STS
 2280 RAR
 2290 JNC BS
 2300 SWAP INX H
 2310 MOV A, M
 2320 DCX H
 2330 MOV M, A
 2340 INX H
 2350 MOV A, L
 2360 ANI 3FH

*CTL I FOR INSERT (SET I)
 *CTL T FOR TEXT (X I/D)
 *CTL F FOR FEED (SET ALF)
 *CTL N FOR NORMAL TTY (X
 *CTL S FOR SCROLL (SET S)
 *CTL P FOR PAGE (X SCRL)
 *LINE FEED
 *CARRIAGE RETURN
 *ANY OTHER CHARACTER
 *HOME CURSOR
 *FORM FEED
 *LINE TERMINATION CHAR 7F
 *SCREEN END?
 *CLEAR, GO HOME
 *CURSOR RIGHT
 *CURSOR UP
 *SET I/D MODE
 *RIGHT BIT =1
 *CURSOR LEFT
 *RUB OUT IF I/D SET
 *DEL CHAR, SWAP LINE IN

1DA8 C2 A0 1D	2370 JNZ SWAP
1DAB 2B	2380 DCX H
1DAC 36 7F	2390 MVI M, 7FH
1DAE C3 6F 1E	2400 JMP OUT1
1DB1 3A FE 1C	2410 RID LDA STS
1DB4 E6 FE	2420 ANI 0FEH
1DB6 32 FE 1C	2430 STA STS
1DB9 C3 6F 1E	2440 JMP OUT1
1DBC 3A FE 1C	2450 SALF LDA STS
1DBF F6 40	2460 ORI 40H
1DC1 32 FE 1C	2470 STA STS
1DC4 C3 6F 1E	2480 JMP OUT1
1DC7 3A FE 1C	2482 RALF LDA STS
1DCA E6 BF	2484 ANI 0BFH
1DCC 32 FE 1C	2486 STA STS
1DCF C3 6F 1E	2488 JMP OUT1
1DD2 3A FE 1C	2490 SSC LDA STS
1DD5 F6 80	2500 ORI 80H
1DD7 32 FE 1C	2510 STA STS
1DDA C3 6F 1E	2520 JMP OUT1
1DDD 3A FE 1C	2530 RSC LDA STS
1DE0 E6 7F	2540 ANI 7FH
1DE2 32 FE 1C	2550 STA STS
1DE5 C3 6F 1E	2560 JMP OUT1
1DE8 21 40 00	2570 LF LXI H, 64
1DEB 19	2580 DAD D
1DEC 3A FE 1C	2590 LDA STS
1DEF 17	2600 RAL
1DF0 DC F9 1D	2610 CC SCRL
1DF3 22 FC 1C	2620 SHLD CURS
1DF6 C3 6F 1E	2630 JMP OUT1
1DF9 7C	2640 SCRL MOV A, H
1DFA FE 04	2650 CPI 4
1DFC D8	2660 RC
1DFD E5	2670 PUSH H
1DFE 11 00 88	2680 LXI D, SCRN
1E01 21 40 88	2700 LXI H, SCRN+LINE
1E04 7E	2710 SWP MOV A, M
1E05 23	2720 INX H
1E06 EB	2730 XCHG
1E07 77	2740 MOV M, A
1E08 23	2760 INX H
1E09 EB	2770 XCHG
1E0A 7C	2780 MOV A, H
1E0B FE 8C	2800 CPI SEND
1E0D C2 04 1E	2810 JNZ SWP
1E10 EB	2812 XCHG
1E11 06 3F	2814 MVI B, LT
1E13 70	2816 LAST MOV M, B
1E14 23	2820 INX H
1E15 7D	2830 MOV A, L
1E16 FE 00	2840 CPI 0
1E18 C2 13 1E	2850 JNZ LAST

*RESET I/O MODE
*RIGHT BIT =0

*SET ALF MODE
*2ND BIT LEFT =1

*RESET ALF MODE
*2ND BIT LEFT =0

*SET SCROLL MODE
*LEFT BIT =1

*RESET SCROLL MODE
*LEFT BIT =0

*LINE FEED
*ADD 64 TO REL CURSOR

*CHECK SCROLL
*UPDATE CURSOR LOCATION

*SCROLL ROUTINE
*OFF PAGE?
*IF NOT, DO NOTHING

*TAKE IT FROM THE TOP

*GRAB CHARACTER

*GET ADDRESS ONE LINE UP
*PUT CHARACTER THERE

*SCREEN FINISHED?
*TAKE NEXT CHAR IF NOT

*BLANK LAST LINE

1E1B E1	2860 POP H	*GET BACK REL CURSOR
1E1C 11 C0 FF	2862 LXI D, 0-LINE	
1E1F 19	2864 DAD D	*MOVE UP ONE LINE
1E20 C9	2870 RET	
1E21 3A FE 1C	2890 CR LDA STS	*CARRIAGE RETURN
1E24 1F	2900 RAR	
1E25 DA 32 1E	2910 JC BACK	*INSERT/DELETE? IF SO, D
1E28 36 3F	2920 SLOP MVI M, LT	*SCRATCH END OF LINE
1E2A 23	2930 INX H	
1E2B 3E 3F	2940 MVI A, 3FH	*MAKE 1FH FOR 32 CHAR LI
1E2D A5	2950 ANA L	
1E2E C2 28 1E	2960 JNZ SLOP	
1E31 2B	2970 DCX H	
1E32 3E C0	2980 BACK MVI A, 0C0H	*GO TO BEGINNING OF LINE
1E34 A3	2990 ANA E	
1E35 5F	3000 MOV E, A	
1E36 3A FE 1C	3020 LDA STS	
1E39 17	3030 RAL	
1E3A 17	3040 RAL	
1E3B DA E8 1D	3050 JC LF	*CHECK AUTO LINE FEED
1E3E EB	3052 XCHG	
1E3F 22 FC 1C	3055 SHLD CURS	
1E42 C3 6F 1E	3060 JMP OUT1	
1E45 3A FE 1C	4000 DEF LDA STS	*DEFAULT ROUTINE, CHECK I
1E48 1F	4010 RAR	
1E49 DC 5C 1E	4020 CC INSR	*INSERT IF NOTED
1E4C 70	4030 MOV M, B	*STUFF CHARACTER
1E4D 13	4040 INX D	*INCREMENT CURSOR
1E4E EB	4050 XCHG	
1E4F 3A FE 1C	4060 LDA STS	
1E52 17	4070 RAL	
1E53 DC F9 1D	4080 CC SCRL	*CHECK SCROLL
1E56 22 FC 1C	4090 SHLD CURS	*UPDATE CURSOR
1E59 C3 6F 1E	4100 JMP OUT1	
1E5C E5	4200 INSR PUSH H	*MAKE SPACE FOR INSERT
1E5D 7E	4210 MOV A, M	
1E5E 3A FF 1C	4220 LDA STR	
1E61 77	4230 MOV M, A	*REPLACE CHAR UNDER CURSOR
1E62 23	4240 SHFT INX H	*MOVE LINE OUT
1E63 4E	4250 MOV C, M	
1E64 77	4260 MOV M, A	
1E65 3E 3F	4270 MVI A, 3FH	
1E67 A5	4280 ANA L	
1E68 79	4290 MOV A, C	
1E69 C2 62 1E	4300 JNZ SHFT	
1E6C 77	4310 MOV M, A	
1E6D E1	4320 POP H	
1E6E C9	4330 RET	
1E6F 2A FC 1C	8000 OUT1 LHLD CURS	*KEEP CURSOR ON SCREEN
1E72 7C	8010 MOV A, H	
1E73 E6 03	8020 ANI 3	

1E75 67
1E76 22 FC 1C
1E79 11 00 88
1E7C 19
1E7D 7E
1E7E 32 FF 1C
1E81 36 FF
1E83 C9

8030 MOV H, A
8040 SHLD CURS
8060 LXI D, SCRN
8070 DAD D
8080 MOV A, M
8090 STA STR
8100 MVI M, CS
8110 RET

*INDEX BY SCREEN ADDRESS

*STORE CHAR UNDER CURSOR

*STUFF NEW CURSOR SYMBOL

Life for the VTI

0000	0100	VADD EQU 8800H	*VIDEO BLOCK ADDRESS
0000	0110	MADD EQU 0300H	*MASTER COPY ADDRESS
0000	0120	SADD EQU 0800H	*SLAVE COPY ADDRESS
0000	0130	MAD EQU 03H	*1ST BYTE OF MADD
0000	0140	SAD EQU 08H	*1ST BYTE OF SADD
0000	0150	LINE EQU 64	*LINE LENGTH
0000	0160	TADD EQU 0208H	*TABLE (MASK & SCRATCH)
0000	0170	TAD EQU 02H	*1ST BYTE OF TADD
0000	0175	CADD EQU 0180H	*COUNT ADDRESS (GENERATION)
0000 21 08 02	0180	LXI H, TADD	*SET UP MASK TABLE
0003 3E 20	0185	MVI A, 20H	*FIRST MASK FOR TABLE
0005 0E 08	0190	MASK MVI C, 08H	*GETS EIGHT SPOTS
0007 77	0200	TABLE MOV M, A	
0008 23	0210	INX H	
0009 0D	0220	DCR C	
000A C2 07 00	0230	JNZ TABLE	*IN TABLE.
000D 0F	0240	RRC	*THEN MASK FOR NEXT LOWER
000E D2 05 00	0250	JNC MASK	*GETS THE NEXT EIGHT.
0011 21 00 08	0254	LXI H, SADD	*SAVE SLAVE ADDRESS
0014 E5	0256	PUSH H	*FOR USE IN LOOP
0015 21 80 01	0258	LXI H, CADD	*LOAD CADD WITH OWN
0018 36 80	0260	MVI M, 80H	*SECOND BYTE TO START COUNT
001A 21 C0 87	0270	LXI H, VADD-40H	*SET UP FOR SWAP FROM
001D 11 C0 07	0280	LXI D, SADD-40H	*SCREEN TO SLAVE WITH SLOP
0020 7E	0282	LOOP MOV A, M	*GRAB CHAR, BEGIN MAIN LOOP
0021 2F	0284	CMA	*COMPLEMENT FOR TRUE LIFE
0022 12	0286	STAX D	*STORE ON OTHER COPY.
0023 23	0288	INX H	*NEXT
0024 13	0290	INX D	*SPOT.
0025 7C	0292	MOV A, H	*CHECK
0026 E6 07	0294	ANI 7	*LAST THREE BITS OF 1ST BY
0028 FE 05	0296	CPI 5	*FOR END
002A C2 20 00	0298	JNZ LOOP	*OF COPY PLUS SLOP.
002D 21 C0 07	0300	LXI H, SADD-40H	*SWAP SLAVE
0030 11 C0 02	0310	LXI D, MADD-40H	*TO MASTER
0033 7E	0312	SWAP MOV A, M	
0034 12	0314	STAX D	
0035 23	0316	INX H	
0036 13	0318	INX D	
0037 7C	0320	MOV A, H	
0038 FE 0D	0324	CPI SAD+5	*WITH SLOP
003A C2 33 00	0326	JNZ SWAP	*UP TO HERE.
003D 11 80 01	0330	LXI D, CADD	*SET UP FOR COUNT
0040 01 40 88	0340	LXI B, VADD+40H	*IN UPPER RIGHT OF SCREEN
0043 21 80 01	0350	LXI H, CADD	*WATCH THE ZERO AND CARRY!
0046 6B	0360	MOV L, E	


```

0047 23      0370 COUNT INX H
0048 0E      0380 DCX B
0049 C2 4D 00 0390 JNZ NOINC
004C 34      0400 INR M
004D 1A      0410 NOINC LDAX D
004E BD      0420 CMP L
004F DA 58 00 0430 JC OUT
0052 3E BA    0440 MYI A, 0BAH
0054 BE      0450 CMP M
0055 C2 5B 00 0460 JNZ HERE
0058 3E B0    0570 OUT MYI A, 0B0H
005A 77      0580 MOV M, A
005B 7E      0590 HERE MOV A, M
005C 02      0600 STAX B
005D D2 47 00 0610 JNC COUNT
0060 2B      0620 DCX H
0061 BE      0630 CMP M
0062 CA 67 00 0640 JZ THERE
0065 EB      0650 XCHG
0066 34      0660 INR M
0067 21 BF 02 1040 THERE LXI H, MADD-LINE-1
006A 16 02    1050 MYI D, TAD
006C 01 D6 00 1060 BYTE LXI B, INST
006F 0A      1090 BIT LDAX B
0070 0F      1100 RRC
0071 D2 87 00 1110 JNC ROT
0074 0F      1120 RRC
0075 D2 83 00 1130 JNC ONE
0078 FE F0    1140 CPI 0F0H
007A D2 99 00 1150 JNC DONE
007D 11 3D 00 1170 LXI D, LINE-3
0080 19      1180 DAD D
0081 16 02    1190 MYI D, TAD
0083 23      1210 ONE INX H
0084 E6 3F    1215 ANI 3FH
0086 07      1220 RLC
0087 1F      1230 ROT RAR
0088 03      1240 INX B
0089 5F      1250 MOV E, A
008A 1A      1260 LDAX D
008B A6      1270 ANA M
008C CA 6F 00 1280 JZ BIT
008F EB      1290 XCHG
0090 3E 07    1300 MYI A, 07H
0092 A5      1310 ANA L
0093 6F      1320 MOV L, A
0094 34      1330 INR M
0095 EB      1340 XCHG

```

```

*NEXT SIGNIFICANT DIGIT
*NEXT DOWN ON SCREEN
*ZERO FLAG TO INCREMENT

*ARE WE TO END
*OF COUNT (STORED AT CADD)?
*YES
*NO, CHECK FOR
*DECIMAL CARRY IN ASCII.
*NO
*YES, ZERO THAT DIGIT
*AND REPLACE MEMORY.
*GET MEMORY
*AND VIEW IT
*UNTIL ALL DIGITS ARE VIEWED.
*CHECK MOST SIGNIFICANT DIGIT
*AGAINST NEXT MOST.
*BOTH ZERO? EXIT.
*NO, INCREASE
*END OF COUNT.

*GET IN POSITION FOR TABLE.
*PSEUDO OP LIST
*LOAD PSEUDO OP.
*CHECK RIGHT BIT FOR
*CELL CHECK FROM SAME BYTE
*NO. NEXT BYTE?
*YES
*NO. ALL NGHBR S DONE THIS BYT
*YES.
*NEXT LINE ON 3X3 MATRIX
*INCREMENT BY LINE-2
*BY LINE-3+1, SINCE WE NEED
*A +1 ANYWAY
*GET RID OF 2 MSB'S.
*ZERO CARRY BIT AND
*GET IN POSITION
*FOR THIS AND NEXT PSEUDO OP
*2ND BYTE FEEDS MASK TABLE
*LOAD MASK FOR BIT
*AND CHECK IT ON THE MASTER
*NO LIFE, NEXT BIT
*BRING DOWN SCRATCH
*ADDRESS TO STORE NEIGHBOR
*COUNT CODED BY BIT

*COUNT ONE NEIGHBOR
*GET MASTER COPY

```


0096 C3 6F 00	1350 JMP BIT	*AND GET NEXT BIT IN BYTE
0099 01 BF FF	1360 DONE LXI B, 0-LINE-1	*GO BACK TO BYTE
009C 09	1370 DAD B	*THAT WE'RE WORKING ON
009D 1E 00	1375 MYI E, 0	*ZERO SCRATCHPAD BYTE #2
009F E3	1380 XTHL	*MOVING ON TO SLAVE COPY.
00A0 97	1390 LOAD SUB A	*ZERO A SO WE CAN
00A1 12	1400 STAX D	*ZERO NEIGHBOR COUNT
00A2 79	1410 MOV A, C	*GET INVERTED BIT MASK
00A3 0F	1420 RRC	*COMING IN BFH AND ROTATE
00A4 D2 BF 00	1430 JNC NEXT	*GOT ALL BITS?
00A7 4F	1440 MOV C, A	*NO, REPLACE MASK
00A8 1C	1450 INR E	*AND COUNT BIT NUMBER
00A9 1A	1460 LDAX D	*GET # NEIGHBORS OF THAT BIT
00AA FE 02	1470 CPI 02H	*IS IT TWO?
00AC CA A0 00	1480 JZ LOAD	*YES, CELL STAYS THE WAY IT
00AF 79	1490 MOV A, C	*NO, SO
00B0 A6	1510 ANA M	*KILL CELL ON
00B1 77	1520 MOV M, A	*SLAVE COPY
00B2 1A	1540 LDAX D	*HOW MANY NHRS AGAIN?
00B3 FE 03	1550 CPI 03H	*ARE THERE THREE?
00B5 C2 A0 00	1560 JNZ LOAD	*YES, GOOD WE KILLED IT.
00B8 79	1570 MOV A, C	*OOPS, GOT TO RESURRECT IT
00B9 2F	1580 CMA	*BY INVERTING THE MASK
00BA 86	1590 ADD M	*AND ADDING
00BB 77	1610 MOV M, A	*REPLACE SLAVE
00BC C3 A0 00	1630 JMP LOAD	*UPDATE NEXT BIT IN BYTE
00BF 01 C0 FF	1640 NEXT LXI B, 0-LINE	*UP ONE, WHICH IS UPPER
00C2 23	1660 INX H	*INCREMENT SLAVE ADDRESS
00C3 E3	1670 XTHL	*FOR PROPER INITIALIZATION
00C4 3E 07	1680 MYI A, MAD+04H	*END OF SCREEN?
00C6 BC	1690 CMP H	
00C7 09	1700 DAD B	
00C8 C2 6C 00	1710 JNZ BYTE	*COMPLETE ONE UP
00CB E1	1715 POP H	*SCREEN NOT OVER, NEXT BYTE
00CC 21 00 08	1720 LXI H, SADD	*LEAVE
00CF E5	1725 PUSH H	*SADD ON STACK
00D0 11 00 88	1740 LXI D, VADD	*FOR NEXT TIME. SET UP TO
00D3 C3 20 00	1830 JMP LOOP	*SWAP SLAVE TO SCREEN
00D6 C4 65	1840 INST DW 65C4H	*ON EACH SUCCESSIVE LOOP.
00D8 C4 70	1850 DW 70C4H	*PSEUDO OPS CODE 48
00DA D0 71	1860 DW 71D0H	*SPECIAL CASES: EIGHT
00DC 87 A4	1870 DW 0A487H	*NEIGHBORS FOR EACH OF
00DE 88 A8	1880 DW 0A888H	*SIX CELLS PER BYTE
00E0 C8 AC	1890 DW 0ACC8H	*RIGHT TWO BITS OF
00E2 CC 45	1900 DW 45CCH	*EACH PSEUDO OP INDICATE
00E4 84 A4	1910 DW 0A484H	*WHETHER NEXT NEIGHBOR IS
00E6 28 68	1920 DW 6828H	*IN THE SAME BYTE AS
00EA 88 A8	1930 DW 0A888H	*CURRENT NEIGHBOR, OR IN
		*NEXT BYTE, OR NEXT LINE

00EA C8 4C	1940 DW 4CC8H
00EC AC CC	1950 DW 0CCACH
00EE 30 50	1960 DW 5030H
00F0 B0 34	1970 DW 34B0H
00F2 54 74	1980 DW 7454H
00F4 94 D4	1990 DW 0D494H
00F6 58 78	2000 DW 7858H
00F8 B8 31	2010 DW 31B8H
00FA 50 34	2020 DW 3450H
00FC 54 74	2030 DW 7454H
00FE 58 78	2040 DW 7858H
0100 8F 2D	2050 DW 2D8FH
0102 8C 38	2060 DW 388CH
0104 98 39	2070 DW 3998H
0106 FF	2080 DB 0FFH

*IN 3X3 MATRIX OF
 *NEIGHBOR BYTES
 *NEXT THREE BITS CODE
 *CELL WHOSE NEIGHBORS
 *WE ARE COUNTING, IN
 *REVERSE ORDER
 *REMAINING THREE BITS
 *CODE MASK FOR NEIGHBOR
 *IN SAME FORMAT

SET SP → 0FFFH

Screen clearing routine

ASSM(CLEAR) 0F00

0F00 21 00 88
 0F03 36 7F
 0F05 23
 0F06 7C
 0F07 FE 8C
 0F09 C2 03 0F
 0F0C 76

1000 LXI H, 8800H
 1010 LOOP MVI M, 7FH
 1020 INX H
 1030 MOV A, H
 1040 CPI 8CH
 1050 JNZ LOOP
 1060 HLT

These programs will take a character from the keyboard when a key is depressed and display it on the screen. The display position will advance 1 character position everytime a key is depressed. Carriage returns and line feeds are not recognized as such and will appear on the screen as Greek letters or special symbols. On 32 character boards the first 32 characters in each line are displayed and the second 32 will not be displayed. When running the normal video driver a carriage return moves the cursor to the beginning of the next line.

If this test does not work, check the keystrobe polarity and your keyboard wiring, otherwise see the troubleshooting section.

This completes the setup of your PolyMorphic Video Terminal Interface. See section 7 for operation of the software supplied with the VTI.

low address

<u>Address</u>	<u>Data</u>	<u>Program</u>
0 ←	F3	DI
1	21	LXI H, 08800H
2	00	
3	88	
4	0C	LOOP: INR C
5	C2	JNZ LOOP
6	04	
7	00	
8	DB	IN 089H
9	89	
A	E6	ANI 1
B	01	
C	C2	JNZ LOOP
D	04	
E	00	
F	DB	IN 080H
10	88	
11	F6	ORI 80H
12	80	
13	77	MOV M, A
14	23	INX H
15	C3	JMP LOOP
16	04	
17	00	

high address

<u>Address</u>	<u>Data</u>	<u>Program</u>
0C80 ←	F3	DI
0C81	21	LXI H, 0F800H
0C82	00	
0C83	F8	
0C84	0C	LOOP: INR C
0C85	C2	JNZ LOOP
0C86	84	
0C87	0C	
0C88	DB	IN 0F9H
0C89	F9	
0C8A	E6	ANI 1
0C8B	01	
0C8C	C2	JNZ LOOP
0C8D	84	
0C8E	0C	
0C8F	DB	IN 0F8H
0C90	F8	
0C91	F6	ORI 80H
0C92	80	
0C93	77	MOV M, A
0C94	23	INX H
0C95	C3	JMP LOOP
0C96	84	
0C97	0C	

PRAGMATIC SYSTEMS
P.O. BOX 43
MOUNTAIN VIEW, CA 94042

UT1 - Utility Subroutine Package

The UT1 subroutine package is a collection of commonly used microcomputer subroutines written in 8080 assembly language. By providing many commonly used I/O routines and code conversions the package helps to reduce the size of source programs. The routines occupy less than 3/4 k of memory, and ideally they will be located in a PROM or EPROM. A branch table at the top of the program provides the jump vectors to the individual routines. This permits the applications programs to use the same call addresses, even if some internal changes have been made to the subroutines. There is room left in this table for the addresses of 10 user defined subroutines.

The standard program is ORG at 0C000H, placing it high in the system memory map. Each of the individual routines has a header defining pass parameters, error conditions, and any side effects.

The subroutines are as follows:

<u>SUBROUTINE</u>	<u>CALL ADDRESS</u>	<u>FUNCTION</u>
CIN	0C000H	Console character input
COUT	0C003H	Console character output
CIO	0C006H	Console input and echo
RIN	0C009H	ASR reader character input
RIPAR	0C00CH	Parallel reader input
POPAR	0C00FH	Parallel punch output
LOPAR	0C012H	Parallel printer output
IN8H	0C015H	Input 8 bit hex number
IN16H	0C018H	Input 16 bit hex number
IN8O	0C01BH	Input 8 bit octal number
IN16O	0C01EH	Input 16 bit octal number
IN8D	0C021H	Input 8 bit decimal number
IN16D	0C024H	Input 16 bit decimal number
BCD2	0C027H	Input 2 BCD digits
BCD4	0C02AH	Input 4 BCD digits
OUT8H	0C02DH	Output 8 bit hex number
OU16H	0C030H	Output 16 bit hex number
OUT8O	0C033H	Output 8 bit octal number
OU16O	0C036H	Output 16 bit octal number
STRIN	0C039H	Input character string
STOUT	0C03CH	Output character string
CRLF	0C03FH	Output CR, LF
ASHEX	0C042H	Convert ASCII byte to Hex nibble
HEXAS	0C045H	Convert hex nibble to ASCII byte
DECHEX	0C048H	Convert BCD Number to hex
HEXDEC	0C04BH	Convert 16 bit hex number to BCD

In addition, there are several internal routines not referenced by the master table which some users may find useful.

I/O PORT ASSIGNMENTS

The UT1 subroutines use 3 input ports and 4 output ports for I/O device status, control, and data. All data is assumed positive true. However, a NOP instruction follows each input instruction and preceeds each output instruction. This allows the user to patch in CMA instructions as required to match the signal polarity of his I/O devices.

The Input ports are assigned as follows:

<u>Name</u>	<u>Address</u>	<u>Function</u>
STIN	0	I/O device status input port
CONIN	1	Console device input port
REIN	2	Parallel reader data input port

The bits in the status input port are assigned as follows:

<u>Bit</u>	<u>Function</u>
0	Console Input Status
1	Console Output Status
2	Reader Input Status
3	Punch Output Status
4	Printer Output Status

The Output ports are assigned as follows:

<u>Name</u>	<u>Address</u>	<u>Function</u>
CTLOUT	0	I/O device control output port
CONOUT	1	Console device output port
PUOUT	2	Parallel punch data output port
LIOUT	3	Parallel printer data output port

The bits in the control output port are assigned as follows:

<u>Bit</u>	<u>Function</u>
0	Relay strobe for ASR reader
1	Read strobe for parallel reader
2	Punch strobe for parallel punch
3	Write strobe for parallel printer

; UT1 - UTILITY SUBROUTINE PACKAGE

; COPYRIGHT (C) 1976, PRAGMATIC SYSTEMS

; P.O BOX 43, MOUNTAIN VIEW, CA 94042

; THIS PROGRAM MAY NOT BE REPRODUCED IN WHOLE OR
; PART BY ANY MEANS WITHOUT THE WRITTEN PERMISSION
; OF PRAGMATIC SYSTEMS.

0000	STIN	EQU 0	; I/O DEVICE STATUS PORT
0001	CONIN	EQU 1	; CONSOLE INPUT PORT
0002	REIN	EQU 2	; READER DATA INPUT PORT
0000	CTLOUT	EQU 0	; I/O DEVICE CONTROL PORT
0001	CONOUT	EQU 1	; CONSOLE OUTPUT PORT
0002	PUDOUT	EQU 2	; PUNCH DATA OUTPUT PORT
0003	LIOUT	EQU 3	; LIST DATA OUTPUT PORT

; MASTER BRANCH TABLE

; UART OR PARALLEL I/O DRIVERS FOR TERMINALS
; (FOR PUNCH AND LIST OUTPUT TO TERMINAL USE C0)

C000		ORG 0C000H	
C000	C36CC0	JMP CIN	; CHARACTER INPUT
C003	C378C0	JMP COUT	; CHARACTER OUTPUT
C006	C385C0	JMP CIO	; CHARACTER INPUT AND ECHO
C009	C392C0	JMP RIN	; ASR READER AND UART

; PARALLEL DEVICE I/O DRIVERS

C00C	C3B4C0	JMP RIPAR	; PARALLEL READER INPUT
C00F	C3EEC0	JMP PDPAR	; PARALLEL PUNCH OUTPUT
C012	C304C1	JMP LOPAR	; PARALLEL PRINTER OUTPUT

; NUMERIC I/O ROUTINES

C015	C31AC1	JMP IN8H	; INPUT 8 BIT HEX NUMBER
C018	C321C1	JMP IN16H	; INPUT 16 BIT HEX NUMBER
C01B	C334C1	JMP IN8O	; INPUT 8 BIT OCTAL NUMBER
C01E	C33BC1	JMP IN16O	; INPUT 16 BIT OCTAL NUMBER
C021	C351C1	JMP IN8D	; INPUT 8 BIT DECIMAL NUMBER
C024	C358C1	JMP IN16D	; INPUT 16 BIT DECIMAL NUMBER
C027	C36DC1	JMP BCD2	; INPUT 2 BCD DIGITS
C02A	C374C1	JMP BCD4	; INPUT 4 BCD DIGITS
C02D	C38AC1	JMP OUT8H	; OUTPUT 8 BIT HEX NUMBER
C030	C3A3C1	JMP OUT16H	; OUTPUT 16 BIT HEX NUMBER
C033	C3AEC1	JMP OUT8O	; OUTPUT 8 BIT OCTAL NUMBER
C036	C3C6C1	JMP OUT16O	; OUTPUT 16 BIT OCTAL NUMBER

; STRING I/O


```

C039 C3F4C1      JMP STRIN      ;INPUT CHARACTER STRING
C03C C33CC2      JMP STOUT      ;OUTPUT CHARACTER STRING
C03F C351C2      JMP CRLF       ;OUTPUT CR,LF

```

```

;
;  NUMBER CONVERSIONS
;

```

```

C042 C360C2      JMP ASHEX      ;CONVERT ASCII BYTE TO HEX NIBBLE
C045 C373C2      JMP HEXAS      ;CONVERT HEX NIBBLE TO ASCII BYTE
C048 C37CC2      JMP DECHEX     ;CONVERT BCD NUMBER TO HEX
C04B C3AFC2      JMP HEXDEC     ;CONVERT 16 BIT HEX NUMBER TO BCD

```

```

001E              DS 30          ;LEAVE ROOM FOR 10 USER ADDRESSES

```

```

;
;  CIN - INPUT SINGLE CHARACTER FROM CONSOLE DEVICE
;

```

```

;  PASS IN:  NOTHING
;  PASS BACK: EIGHT BIT CHARACTER IN A
;  MODIFIES: A AND ALL FLAGS
;

```

```

C06C DB00      CIN:  IN STIN      ;CHECK STATUS
C06E 00        NOP              ;IN CASE CMA NEEDED
C06F E601      ANI 1
C071 CA6CC0    JZ CIN           ;WAIT FOR READY
C074 DB01      IN CONIN        ;READ CHARACTER
C076 00        NOP              ;IN CASE CMA NEEDED
C077 C9        RET

```

```

;
;  COUT - OUTPUT BYTE TO CONSOLE DEVICE
;

```

```

;  PASS IN:  BYTE TO BE OUTPUT IN C
;  PASS BACK: NOTHING
;  MODIFIES: A AND ALL FLAGS
;

```

```

C078 DB00      COUT:  IN STIN      ;CHECK STATUS
C07A 00        NOP              ;IN CASE CMA NEEDED
C07B E602      ANI 2
C07D CA78C0    JZ COUT         ;WAIT FOR READY
C080 79        MOV A,C         ;GET BYTE
C081 00        NOP              ;IN CASE CMA NEEDED
C082 D301      OUT CONOUT
C084 C9        RET

```

```

;
;  CIO - INPUT AND ECHO TTY COMMAND FROM UART
;

```

```

;  PASS IN:  NOTHING
;  PASS BACK: CHARACTER IN A WITH BIT 7 SET TO 0
;  MODIFIES: A AND ALL FLAGS
;

```

```

C085 C5        CIO:  PUSH B
C086 CD6CC0    CALL CIN         ;GET CHARACTER
C089 E67F      ANI 7FH         ;REMOVE PARITY BIT

```



```

C08B 4F          MOV C,A
C08C CD78C0      CALL COUT          ;ECHO
C08F 79          MOV A,C          ;RESTORE CHARACTER
C090 C1          POP B
C091 C9          RET

;
; RIN - INPUT CHARACTER FROM ASR 33 WITH RELAY CONTROL
;
; PASS IN:  NOTHING
; PASS BACK: CHARACTER IN A
; ERROR:    CARRY SET IF NO CHARACTER IN 1 SEC.
; MODIFIES: A AND ALL FLAGS
;
C092 3E01      RIN:  MVI A,1          ;STROBE RELAY
C094 00        NOP                  ;IN CASE CMA NEEDED
C095 D300      OUT CTOUT            ;OUTPUT TO CONTROL PORT
C097 AF        XRA A
C098 00        NOP                  ;IN CASE CMA NEEDED
C099 D300      OUT CTOUT

;
; RELAY STROBED. WAIT FOR THE DATA
;
C09B C5        PUSH B
C09C 06C8      MVI B,200            ;SET TIME OUT COUNTER
C09E DB00      RI1:  IN STIN
COA0 E601      ANI 1                ;SEE IF TTY DATA READY
COA2 C2AFCD    JNZ RI2
COA5 CDE0C0    CALL D05             ;DELAY 10 MS
COA8 05        DCR B
COA9 C29EC0    JNZ RI1             ;IF NOT ZERO, KEEP WAITING
COAC C3D7C0    JMP RITRM           ;IF ZERO, CONSIDER TIMED OUT

;
; READ IN DATA FROM TTY INPUT PORT
;
COAF DB01      RI2:  IN CONIN        ;GET THE DATA
COB1 00        NOP                  ;IN CASE CMA NEEDED
COB2 C1        POP B
COB3 C9        RET

;
; RIPAR - PARALLEL READER INPUT
;
; PASS IN:  NOTHING
; PASS BACK: CHARACTER IN A
; ERROR:    CARRY SET IF NO CHARACTER IN 1 SEC
; MODIFIES: A AND ALL FLAGS
;
COB4 3E02      RIPAR: MVI A,2        ;STROBE STEPPER MOTOR
COB6 00        NOP                  ;IN CASE CMA NEEDED
COB7 D300      OUT CTOUT
COB9 AF        XRA A
COBA 00        NOP                  ;IN CASE CMA NEEDED

```



```

COBB D300 OUT CTLOUT
;
; MOTOR STROBED. WAIT FOR FEED HOLE TO TRANSITION
;
COBD DB00 RIP1: IN STIN ;MAKE SURE HOLE DARK
COBF 00 NOP ;IN CASE CMA NEEDED
COC0 E604 ANI 4
COC2 C2BDC0 JNZ RIP1 ;LOOP UNTIL PREVIOUS HOLE GONE
;
; WAIT 1 SEC. FOR DATA
;
COC5 C5 PUSH B
COC6 06C8 MVI B,200
COC8 DB00 RIP2: IN STIN
COCA 00 NOP ;IN CASE CMA NEEDED
COCB E604 ANI 4 ;WAIT FOR HOLE
COCd C2DBC0 JNZ RIP3
COD0 CDE0C0 CALL D05
COD3 05 DCR B
COD4 C2C8C0 JNZ RIP2
;
; 1 SECOND WITH NO DATA. TREAT AS EOF
;
COD7 AF RITRM: XRA A ;CLEAR RETURN REGISTER
COD8 37 STC ;SET FLAG
COD9 C1 POP B
CODA C9 RET
;
; INPUT DATA
;
COdB DB02 RIP3: IN REIN ;BAG THE READER DATA
CODD 00 NOP ;IN CASE CMA NEEDED
CODE C1 POP B
CODF C9 RET
;
; D05 - DELAY 5 MS
; THIS ROUTINE PROVIDES APPRX. 5 MS OF DELAY
; WITH A 2MH CLOCK AND MEMORIES REQUIRING NO WAITS
; NOT EXTERNALLY REFERENCED
;
COE0 C5 D05: PUSH B
COE1 F5 PUSH PSW
COE2 01B101 LXI B,433
COE5 0B D051: DCX B
COE6 47 MOV B,A
COE7 B1 ORA C
COE8 C2E5C0 JNZ D051
COEB F1 POP PSW
COEC C1 POP B
COED C9 RET
;

```


; POPAR - PARALLEL PUNCH OUTPUT

; PASS IN: BYTE TO BE PUNCHED IN C

; PASS BACK: NOTHING

; MODIFIES: A AND ALL FLAGS

;

COEE	DB00	POPARG	IN STIN	;MAKE SURE PUNCH IS READY
COF0	00		NOP	;IN CASE CMA NEEDED
COF1	E608		ANI 8	
COF3	CAEEC0		JZ POPAR	;WAIT UNTIL PUNCH READY

;

; PUNCH READY. OUTPUT DATA

;

COF6	79		MOV A,C	;GET DATA
COF7	00		NOP	;IN CASE CMA NEEDED
COF8	D302		OUT PUOUT	;OUTPUT TO PUNCH DATA PORT

;

; DATA OUT. STROBE PUNCH

;

COFA	3E04		MVI A,4	
COFC	00		NOP	;IN CASE CMA NEEDED
COFD	D300		OUT CTLOUT	
COFF	AF		XRA A	
C100	00		NOP	;IN CASE CMA NEEDED
C101	D300		OUT CTLOUT	
C103	C9		RET	

;

; LOPAR - OUTPUT BYTE TO PARALLEL LIST DEVICE

;

; PASS IN: BYTE TO BE PRINTED IN C

; PASS BACK: NOTHING

; MODIFIES: A AND ALL FLAGS

;

C104	DB00	LOPAR:	IN STIN	;CHECK FOR PRINTER READY
C106	00		NOP	;IN CASE CMA NEEDED
C107	E610		ANI 10H	
C109	CA04C1		JZ LOPAR	;WAIT UNTIL PRINTER READY

;

; PRINTER READY. OUTPUT DATA

;

C10C	79		MOV A,C	;GET THE DATA
C10D	00		NOP	;IN CASE CMA NEEDED
C10E	D303		OUT LIOUT	;OUTPUT DATA

;

; DATA IN OUTPUT PORT. STROBE PRINTER

;

C110	3E08		MVI A,8	
C112	00		NOP	;IN CASE CMA NEEDED
C113	D300		OUT CTLOUT	
C115	AF		XRA A	
C116	00		NOP	;IN CASE CMA NEEDED

C117 D300 OUT CTLOUT
C119 C9 RET

; IN8H - INPUT 8 BIT HEX NUMBER. (CR) ENDS INPUT

; PASS IN: NOTHING

; PASS BACK: 8 BIT NUMBER IN A

; ERROR: CARRY SET IF USER ENTERED CHARACTER
OTHER THAN 0-9, A-F

; MODIFIES: A AND ALL FLAGS

C11A E5 IN8H: PUSH H
C11B CD21C1 CALL IN16H ;GET NUMBER
C11E 7D MOV A,L ;USE LAST 2 DIGITS ENTERED
C11F E1 POP H
C120 C9 RET

; IN16H - INPUT 16 BIT NUMBER. (CR) ENDS INPUT

; PASS IN: NOTHING

; PASS BACK: 16 BIT NUMBER IN HL

; ERROR: CARRY SET IF USER INPUTS ANY CHARACTER
OTHER THAN 0-9, A-F.

; MODIFIES: A, HL, AND ALL FLAGS

C121 210000 IN16H: LXI H,0 ;CLEAR HL FOR ADDRESS ACCUMULATOR
C124 CD85C0 IN16I: CALL C10 ;GET AND ECHO CHARACTER
C127 FE0D CPI 0DH ;DONE?
C129 C8 RZ
C12A CD60C2 CALL ASHEX ;CONVERT TO 4 BIT HEX
C12D D8 RC
C12E CDA6C2 CALL HL16 ;MULTIPLY HL * 16 AND ADD A
C131 C324C1 JMP IN16I

; IN8O - INPUT 8 BIT OCTAL NUMBER. (CR) ENDS INPUT

; PASS IN: NOTHING

; PASS BACK: 8 BIT NUMBER IN A

; ERROR: CARRY SET IF CHARACTER OTHER THAN 0-7

; MODIFIES: A AND ALL FLAGS

C134 E5 IN8O: PUSH H
C135 CD3BC1 CALL IN16O
C138 7D MOV A,L ;GET LSB
C139 E1 POP H
C13A C9 RET

; IN16O - INPUT 16 BIT OCTAL NUMBER. (CR) ENDS INPUT

; PASS IN: NOTHING

; PASS BACK: 16 BIT NUMBER IN HL


```

; ERROR:      CARRY SET IF CHARACTER OTHER THAN 0-7
; MODIFIES:   A, HL, ALL FLAGS
;

```

```

C13B  210000  IN16D:  LXI H,0      ;CLEAR ADDRESS ACCUMULATOR,
C13E  CD85C0  IN01:   CALL C10
C141  FE0D    CPI 0DH      ;TEST FOR DONE
C143  C8      RZ
C144  FE38    CPI 38H      ;TEST FOR > 7
C146  3F      CMC
C147  D8      RC
C148  D630    SUI 30H      ;SCALE DOWN TO 0-7
C14A  D8      RC
C14B  CDA7C2  CALL HL08    ;MULTIPLY HL * 8 AND ADD A
C14E  C33EC1  JMP IN01

```

```

;
; IN8D - INPUT 8 BIT NUMBER AS DECIMAL DIGITS
;

```

```

; PASS IN:    NOTHING
; PASS BACK:  8 BIT NUMBER IN A
; ERROR:      CARRY SET IF CHARACTER OTHER THAN 0-9
; MODIFIES:   A AND ALL FLAGS
;

```

```

C151  E5      IN8D:  PUSH H
C152  CD58C1  CALL IN16D
C155  7D      MOV A,L      ;GET LSD
C156  E1      POP H
C157  C9      RET

```

```

;
; IN16D - INPUT 16 BIT NUMBER AS DECIMAL DIGITS
;

```

```

; PASS IN:    NOTHING
; PASS BACK:  16 BIT NUMBER IN HL
; ERROR:      CARRY SET IF CHARACTER OTHER THAN 0-9
; MODIFIES:   A, HL, AND ALL FLAGS
;

```

```

C158  210000  IN16D:  LXI H,0
C15B  CD85C0  IND1:   CALL C10      ;GET CHARACTER
C15E  FE0D    CPI 0DH      ;EXIT ON CR
C160  C8      RZ
C161  FE3A    CPI 3AH      ;TEST FOR > 9
C163  3F      CMC
C164  D8      RC
C165  D630    SUI 30H      ;SCALE DOWN TO 0-9
C167  CD97C2  CALL HL10    ;MULTIPLY HL BY 10 AND ADD A
C16A  C35BC1  JMP IND1

```

```

;
; BCD2 - INPUT 2 BCD DIGITS
;

```

```

; PASS IN:    NOTHING
; PASS BACK:  2 PACKED BCD DIGITS IN A
; ERROR:      CARRY SET IF CHARACTER OTHER THAN 0-9

```



```

; MODIFIES: A AND ALL FLAGS
;
C16D E5 BCD2: PUSH H
C16E CD74C1 CALL BCD4
C171 7D MOV A,L ;USE 2 LSD
C172 E1 POP H
C173 C9 RET
;
; BCD4 - INPUT 4 BCD DIGITS
;
; PASS IN: NOTHING
; PASS BACK: 4 PACKED BCD DIGITS IN HL
; ERROR: CARRY SET IF CHARACTER OTHER THAN 0-9
; MODIFIES: A, HL, AND ALL FLAGS
;
C174 210000 BCD4: LXI H,0 ;CLEAR ADDRESS ACCUMULATOR
C177 CD85C0 BCD41: CALL C10 ;GET CHARACTER
C17A FE0D CPI 0DH ;TEST FOR CARRIAGE RETURN
C17C C8 RZ
C17D FE3A CPI 3AH ;TEST FOR > ASCII 9
C17F 3F CMC
C180 D8 RC
C181 D630 SUI 30H ;SCALE DOWN TO 0-9
C183 D8 RC
C184 CDA6C2 CALL HL16 ;SHIFT HL 4 LEFT AND ADD A
C187 C377C1 JMP BCD41
;
; OUT8H - DISPLAY 8 BIT DATA AS 2 HEX CHARACTERS
; (ALSO WORKS FOR 2 BCD DIGITS PACKED IN A)
;
; PASS IN: DATA TO BE DISPLAYED IN A
; PASS BACK: NOTHING
; MODIFIES: NOTHING
;
C18A F5 OUT8H: PUSH PSW
C18B C5 PUSH B
C18C 47 MOV B,A ;SAVE DATA BYTE
C18D 0F RRC ;SHIFT MSB INTO POSITION
C18E 0F RRC
C18F 0F RRC
C190 0F RRC
C191 CD73C2 CALL HEXAS ;CONVERT TO ASCII
C194 4F MOV C,A
C195 CD78C0 CALL COUT ;PRINT IT
C198 78 MOV A,B ;RECOVER DIGIT
C199 CD73C2 CALL HEXAS ;CONVERT LSB TO ASCII
C19C 4F MOV C,A
C19D CD78C0 CALL COUT ;PRINT IT
C1A0 C1 POP B
C1A1 F1 POP PSW
C1A2 C9 RET

```



```

;
;  OUI6H - PRINT 16 BIT DATA AS 4 HEX CHARACTERS
;          (ALSO WORKS FOR 4 BCD DIGITS PACKED IN HL)
;

```

```

;  PASS IN:  16 BIT DATA IN HL
;  PASS BACK: NOTHING
;  MODIFIES:  NOTHING
;

```

```

CIA3  F5      OUI6H:  PUSH PSW
CIA4  7C              MOV A,H          ;OUTPUT 2 MSD
CIA5  CD8AC1      CALL OUT8H
CIA8  7D              MOV A,L          ;OUTPUT 2 LSD
CIA9  CD8AC1      CALL OUT8H
CIAC  F1              POP PSW
CIAD  C9              RET

```

```

;
;  OUT80 - OUTPUT 8 BIT DATA AS 3 OCTAL CHARACTERS
;

```

```

;  PASS IN:  8 BIT DATA VALUE IN A
;  PASS BACK: NOTHING
;  MODIFIES:  NOTHING
;

```

```

CIAE  C5      OUT80:  PUSH B
CIAF  F5              PUSH PSW
CIB0  47              MOV B,A          ;SAVE CHARACTER
CIB1  07              RLC              ;SHIFT MSD INTO POSITION
CIB2  07              RLC
CIB3  E603      OUT81: ANI 3          ;FORCE MSB TO 0
CIB5  CDECC1      CALL MSKOT          ;OUTPUT MSD
CIB8  78              MOV A,B
CIB9  0F              RRC              ;SHIFT MIDDLE DIGIT INTO POSITION
CIBA  0F              RRC
CIBB  0F              RRC
CIBC  CDECC1      CALL MSKOT          ;OUTPUT DIGIT 2
CIBF  78              MOV A,B
CIC0  CDECC1      CALL MSKOT          ;OUTPUT LSD
CIC3  C1              POP B
CIC4  F1              POP PSW
CIC5  C9              RET

```

```

;
;  OUI60 - OUTPUT 16 BIT OCTAL NUMBER
;

```

```

;  PASS IN:  16 BIT OCTAL NUMBER IN HL
;  PASS BACK: NOTHING
;  MODIFIES:  NOTHING
;

```

```

CIC6  C5      OUI60:  PUSH B
CIC7  F5              PUSH PSW
CIC8  7C              MOV A,H
CIC9  07              RLC              ;GET MSD
CICA  E601      ANI 1          ;REMOVE BITS 1 AND 2

```



```

C1CC  CDECC1      CALL MSKOT      ;OUTPUT
C1CF  7C          MOV A,H
C1D0  0F          RRC              ;GET DIGIT 2 INTO POSITION
C1D1  0F          RRC
C1D2  0F          RRC
C1D3  0F          RRC
C1D4  CDECC1      CALL MSKOT      ;OUTPUT DIGIT 2
C1D7  7C          MOV A,H
C1D8  0F          RRC              ;GET DIGIT 3 INTO POSITION
C1D9  CDECC1      CALL MSKOT      ;OUTPUT DIGIT 3
C1DC  7C          MOV A,H
C1DD  07          RLC              ;PREPARE DIGIT 4
C1DE  07          RLC
C1DF  E604        ANI 4            ;MSB READY
C1E1  47          MOV B,A         ;SAVE MSB OF DIGIT 4
C1E2  7D          MOV A,L
C1E3  07          RLC
C1E4  07          RLC              ;GET BITS 2 AND 3 OF DIGIT FOUR
C1E5  E603        ANI 3
C1E7  B0          ORA B           ;ADD MSB
C1E8  45          MOV B,L         ;MATCH UP FOR OUT80
C1E9  C3B5C1      JMP OUT81

```

```

;
; MSKOT - OUTPUT OCTAL DIGIT
; ROUTINE NOT EXTERNALLY REFERENCED
;

```

```

C1EC  E607      MSKOT: ANI 7      ;REMOVE BITS 3-7
C1EE  F630      ORI 30H          ;CONVERT TO ASCII
C1F0  4F        MOV C,A
C1F1  C378C0    JMP COUT        ;OUTPUT

```

```

;
; STRIN - INPUT CHARACTER STRING INTO MEMORY BUFFER
;
; PASS IN: ADDRESS OF MEMORY BUFFER IN HL
; PASS BACK: INPUT STRING IN BUFFER
;          NUMBER OF CHARACTERS INPUT IN A
; ERROR: CARRY SET IF BUFFER FULL ON RETURN
; MODIFIES: A AND ALL FLAGS
;
; UP TO 256 CHARACTERS PERMITTED IN EACH STRING
; ONE CHARACTER DELETED FOR EACH RUBOUT (7FH)
; ENTIRE LINE DELETED BY CNTL C (03H)
; STRING INPUT TERMINATES WITH CR (0DH)
; CONTROL CHARACTERS < 20H ARE IGNORED
; ALL LINES RETURN TERMINATED WITH 0FFH
;

```

```

C1F4  C5      STRIN: PUSH B
C1F5  E5      PUSH H
C1F6  E1      STR1: POP H      ;GET START OF BUFFER
C1F7  E5      PUSH H          ;RE-SAVE IT
C1F8  CD51C2  CALL CRLF       ;START ON FRESH LINE

```


C1FB	0600		MVI B,0	;SET CHARACTER COUNTER
C1FD	CD6CC0	STR2:	CALL CIN	;GET CHARACTER
C200	E67F		ANI 7FH	;REMOVE PARITY
C202	FE03		CPI 3	;TEST FOR LINE RE-START
C204	CAF6C1		JZ STR1	
C207	FE0D		CPI 0DH	;TEST FOR END OF LINE
C209	C212C2		JNZ STR3	
C20C	36FF		MVI M,0FFH	;PLACE LINE TERMINATOR
C20E	78		MOV A,B	;GET COUNT INTO A
C20F	E1		POP H	;RESTORE REGISTERS AND RETURN
C210	C1		POP B	
C211	C9		RET	
C212	FE7F	STR3:	CPI 7FH	;TEST FOR CHARACTER DELETE
C214	C226C2		JNZ STR4	
C217	78		MOV A,B	;TEST FOR 7FH AS FIRST CHARACTER
C218	B7		ORA A	
C219	CAFDC1		JZ STR2	;IF SO, IGNORE IT
C21C	05		DCR B	;DECREMENT CHARACTER COUNTER
C21D	2B		DCX H	;DECREMENT MEMORY POINTER
C21E	0E3C		MVI C,3CH	;ECHO A < AS CHARACTER DELETE
C220	CD78C0		CALL COUT	
C223	C3FDC1		JMP STR2	
C226	FE20	STR4:	CPI 20H	;TEST FOR CONTROL CHARACTERS
C228	FAFDC1		JM STR2	;IF < 20H, IGNORE THEM
C22B	77		MOV M,A	;SAVE CHARACTER IN BUFFER
C22C	23		INX H	;INCREMENT BUFFER POINTER
C22D	4F		MOV C,A	;ECHO CHARACTER
C22E	CD78C0		CALL COUT	
C231	04		INR B	;INCREMENT CHARACTER COUNTER
C232	C2FDC1		JNZ STR2	;IF NOT ZERO, CONTINUE
C235	36FF		MVI M,0FFH	;TERMINATE FULL BUFFER
C237	E1		POP H	
C238	C1		POP B	
C239	AF		XRA A	;SET A TO 0
C23A	37		STC	
C23B	C9		RET	

```

;
;  STOUT - OUTPUT STRING TO CONSOLE
;

```

```

;  PASS IN:  ADDRESS OF STRING IN HL
;             (ALL STRINGS END IN 0FFH)
;  PASS BACK: NOTHING
;  MODIFIES: NOTHING
;

```

C23C	E5	STOUT:	PUSH H	
C23D	C5		PUSH B	
C23E	F5		PUSH PSW	
C23F	4E	STO1:	MOV C,M	;GET CHARACTER
C240	79		MOV A,C	
C241	FEFF		CPI 0FFH	;TEST FOR END OF STRING
C243	CA4DC2		JZ STEND	


```

C246 CD78C0      CALL COUT      ;PRINT CHARACTER
C249 23          INX H
C24A C33FC2      JMP STO1
C24D F1          STEND: POP PSW
C24E C1          POP B
C24F E1          POP H
C250 C9          RET

```

```

;
; CRLF - OUTPUT CR AND LF TO CONSOLE
;

```

```

; PASS IN:  NOTHING
; PASS BACK: NOTHING
; MODIFIES: NOTHING
;

```

```

C251 C5          CRLF:  PUSH B
C252 F5          PUSH PSW
C253 0E0D        MVI C,0DH
C255 CD78C0      CALL COUT      ;PRINT CARRIAGE RETURN
C258 0E0A        MVI C,0AH
C25A CD78C0      CALL COUT      ;PRINT LINE FEED
C25D F1          POP PSW
C25E C1          POP B
C25F C9          RET

```

```

;
; ASHEX - CONVERT ASCII BYTE TO HEX NIBBLE
;
; PASS IN:  ASCII 0-9, A-F IN A
; PASS BACK: HEX NIBBLE IN RANGE 0-F IN A
; ERROR:    CARRY SET IF CHARACTER OTHER THAN 0-9, A-F
; MODIFIES: A AND ALL FLAGS
;

```

```

C260 FE47        ASHEX: CPI 47H      ;TEST FOR > ASCII F
C262 3F          CMC
C263 D8          RC
C264 FE30        CPI 30H      ;TEST FOR < ASCII 0
C266 D8          RC          ;RETURN WITH IT UNMODIFIED
C267 D630        SUI 30H      ;SCALE DOWN
C269 FE0A        CPI 10      ;TEST FOR 0-9
C26B 3F          CMC
C26C F8          RM          ;ALL DONE
C26D D611        SUI 11H      ;FILTER MIDDLE CHARACTERS OUT
C26F D8          RC
C270 C60A        ADI 10      ;CONVERT A-F
C272 C9          RET

```

```

;
; HEXAS - CONVERT HEX NIBBLE TO ASCII BYTE
;

```

```

; PASS IN:  HEX 0-F IN A
; PASS BACK: ASCII 0-F IN A
; MODIFIES: A AND ALL FLAGS
;

```



```

C273 E60F    HEXAS: ANI 0FH          ;STRIP MSB
C275 C690    ADI 90H
C277 27      DAA
C278 CE40    ACI 40H
C27A 27      DAA
C27B C9      RET

```

```

;
; DECHEX - CONVERT BCD NUMBER TO HEX
;
; PASS IN:  LOW ADDRESS OF BCD DIGIT BUFFER IN DE
;            DE POINTS TO MSD
;            C = NUMBER OF DIGITS TO CONVERT (1-5)
; PASS BACK: 16 BIT HEX NUMBER IN HL
; ERROR:     CARRY SET IF NUMBER IN HL EXCEEDS 65,536
; MODIFIES:  HL AND CARRY FLAG
;

```

```

C27C D5      DECHX: PUSH D
C27D C5      PUSH B
C27E F5      PUSH PSW
C27F 210000  LXI H,0          ;CLEAR ACCUMULATOR
C282 1A      DECH1: LDAX D      ;BAG DIGIT
C283 CD97C2  CALL HL10        ;MULTIPLY HL * 10 AND ADD A
C286 13      INX D
C287 0D      DCR C
C288 C282C2  JNZ DECH1        ;CONTINUE UNTIL DONE
C28B DA92C2  JC DECH2         ;IF CARRY, MODIFY PSW RETURN
C28E F1      POP PSW
C28F C1      POP B
C290 D1      POP D
C291 C9      RET
C292 F1      DECH2: POP PSW
C293 37      STC
C294 C1      POP B
C295 D1      POP D
C296 C9      RET

```

```

;
; HL10 - MULTIPLY HL BY 10 AND ADD A
; NOT EXTERNALLY REFERENCED
;

```

```

C297 D5      HL10:  PUSH D
C298 54      MOV D,H
C299 5D      MOV E,L          ;SAVE HL IN DE
C29A 29      DAD H            ;HL = HL * 2
C29B 29      DAD H            ;HL = HL * 4
C29C 19      DAD D            ;HL = HL * 5
C29D 29      DAD H            ;HL = HL * 10
C29E 85      ADD L
C29F 6F      MOV L,A
C2A0 D2A4C2  JNC HL101
C2A3 24      INR H
C2A4 D1      HL101: POP D

```


C2A5 C9 RET

```

;
; HL16 - MULTIPLY HL BY 16 AND ADD A
; HL08 - MULTIPLY HL BY 8 AND ADD A
; NOT EXTERNALLY REFERENCED
;

```

```

C2A6 29 HL16: DAD H ;HL = HL * 2
C2A7 29 HL08: DAD H ;HL = HL * 4, HL = HL * 2
C2A8 29 DAD H ;HL = HL * 8, HL = HL * 4
C2A9 29 DAD H ;HL = HL * 16, HL = HL * 8
C2AA 85 ADD L
C2AB 6F MOV L,A
C2AC D0 RNC
C2AD 24 INR H
C2AE C9 RET

```

```

;
; HEXDEC - CONVERT 16 BIT HEX NUMBER TO DECIMAL
;
; PASS IN: 16 BIT NUMBER IN HL
;          LOW ADDRESS OF 5 BYTE BUFFER IN DE
; PASS BACK: BCD NUMBER IN BUFFER, MSD AT LOW ADDRESS
; MODIFIES: NOTHING
;

```

```

C2AF E5 HEXDEC: PUSH H ;SAVE EVERYTHING
C2B0 D5 PUSH D
C2B1 C5 PUSH B
C2B2 F5 PUSH PSW
C2B3 EB XCHG ;SWAP ADDRESS AND NUMBER
C2B4 011027 LXI B,10000 ;DO MSD
C2B7 CDD7C2 CALL DIG
C2BA 01E803 LXI B,1000
C2BD CDD7C2 CALL DIG
C2C0 016400 LXI B,100
C2C3 CDD7C2 CALL DIG
C2C6 010A00 LXI B,10
C2C9 CDD7C2 CALL DIG
C2CC 010100 LXI B,1
C2CF CDD7C2 CALL DIG
C2D2 F1 POP PSW ;RESTORE EVERYTHING
C2D3 C1 POP B
C2D4 D1 POP D
C2D5 E1 POP H
C2D6 C9 RET

```

```

;
; DIG - ACCUMULATE BCD DIGIT IN M(HL)
; NOT EXTERNALLY REFERENCED
;

```

```

C2D7 3600 DIG: MVI M,0 ;CLEAR THE DIGIT
C2D9 7B DIG1: MOV A,E ;SUBTRACT BC FROM DE
C2DA 91 SUB C
C2DB 5F MOV E,A

```

HL=M

16 bit # in DE

DE=DE-BC

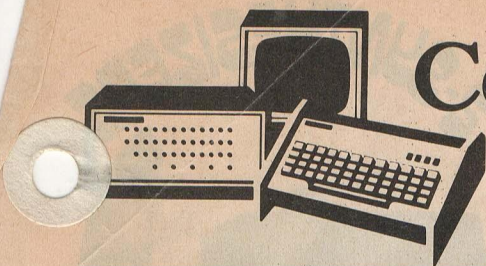
C2DC	7A		MOV A,D	
C2DD	98		SBB B	
C2DE	57		MOV D,A	
C2DF	DAE6C2		JC DIG2	;IF CARRY, DONE
C2E2	34		INR M	
C2E3	C3D9C2		JMP DIG1	
C2E6	EB	DIG2:	XCHG	;CORRECT FOR OVER-RUN
C2E7	09		DAD B	
C2E8	EB		XCHG	
C2E9	23		INX H	;BUMP MEMORY POINTER
C2EA	C9		RET	
			END	

NO PROGRAM ERRORS

SYMBOL TABLE

* 01

A	0007	ASHEX	C260	B	0000	BCD2	C16D
BCD4	C174	BCD41	C177	C	0001	CIN	C06C
CIO	C085	CONIN	0001	CONOU	0001	COUT	C078
CRLF	C251	CTLOU	0000	D	0002	D05	C0E0
D051	C0E5	DECH1	C282	DECH2	C292	DECHE	C27C
DIG	C2D7	DIG1	C2D9	DIG2	C2E6	E	0003
H	0004	HEXAS	C273	HEXDE	C2AF	HL08	C2A7
HL10	C297	HL101	C2A4	HL16	C2A6	IN161	C124
IN16D	C158	IN16H	C121	IN160	C138	IN8D	C151
IN8H	C11A	IN80	C134	IND1	C15B	INO1	C13E
L	0005	LIOUT	0003	LOPAR	C104	M	0006
MSKOT	C1EC	OU16H	C1A3	OU160	C1C6	OUT81	C1B5
OUT8H	C18A	OUT80	C1AE	POPAR	C0EE	PSW	0006
PUDUT	0002	REIN	0002	RI1	C09E	RI2	C0AF
RIN	C092	RIP1	C0BD	RIP2	C0C8	RIP3	C0DB
RIPAR	C0B4	RITRM	C0D7	SP	0006	STEND	C24D
STIN	0000	STO1	C23F	STOUT	C23C	STR1	C1F6
STR2	C1FD	STR3	C212	STR4	C226	STRIN	C1F4



Computer Bits

Computer Music

By Hal Chamberlain

COMPUTER MUSIC

BEGINNING with this month, "Computer Bits" is a monthly feature of POPULAR ELECTRONICS. As a consequence, we are able to offer a two-part discussion of computer music—this month and in October.

Playing music with your computer can be a refreshing diversion, an impressive demonstration, or a mild obsession, depending on your inclination. We will discuss computer music performance first, with computer music composition to follow.

Some History. Elementary, monophonic music is just a series of tones with different frequencies and durations. Over the years, several methods have been devised by which a computer could either generate the tones directly or control tone-generating devices.

Perhaps the strangest method of output is one employed by an IBM-1401 computer program that played the line printer! Computer printers work much like a typewriter in that a metal type face strikes the paper

under computer control. A pulse of sound is emitted when the paper is struck. In a 1401, the printer was interfaced in such a way that a program could control precisely when a print hammer fired. Proper relative timing among hammer strikes in the row of 120 hammers could actually produce a tone. Different frequency tones could be produced by changing the hammer timing. A rendition of "Anchors Aweigh" was heard by the author; and, while sounding rather raspy, it was in tune!

Another unexpected way to get a computer to produce tones involves the use of a common AM broadcast-band radio. Modern computer logic circuits generate pulses having risetimes of about 10 nanoseconds. These fast risetimes imply the existence of high-frequency harmonics that may fall into the radio's tuning range. Furthermore, the harmonic frequencies are separated from each other by the pulse repetition rate. An AM receiver interprets the harmonics within its passband as a carrier with

upper and lower sidebands corresponding to audio modulation at the pulse repetition rate. Different instructions cause pulses to appear at different places within the computer. It is therefore possible to set up program loops that execute at controlled rates and thus produce tones of different frequencies. An interesting problem with this method is that there is no way to get the radio to "shut up" for a rest or a break between two identical notes.

A more predictable way to get audio output is to connect a speaker to an output port bit. A tone may be programmed by alternately setting the bit on and off with an accurately timed loop. This has an advantage since the speaker may be silenced when desired. One commercial computer (the LINC) even had a speaker complete with volume control connected to one bit of the accumulator as standard equipment! Some hobbyist systems have a speaker installed in their surplus keyboards. It is normally clicked to signal that a character has been accepted or beeped if an error is detected. Of course, it is also available as a simple "music peripheral."

Although the above methods make interesting demonstrations and do not require any specialized hardware, they are far from being suitable for serious music performance. Early in the development of electronic music, it became apparent that any audio signal waveform could be represented as a very rapid series of discrete voltage values. With a digital-to-analog converter connected to the computer,

Fig. 1. Example of a tone generation subroutine for the 8080.

16 BIT ADDR 0000H INSTR

				ENTER WITH THE FREQUENCY PARAMETER IN C				
				ENTER WITH THE DURATION PARAMETER IN D & E				CLOCK
				USES SPEAKER CONNECTED TO ANY BIT OF OUTPUT PORT				CYCLES
000:100	076	000	TONE	MVI A,0	OUTPUT ZERO TO SPEAKER			(7)
000:102	232	XXX		OUT (port address)				(10)
000:104	101			MOV B,C	MOVE FREQ. PARAMETER TO			(5)
000:105	005		DELAY1	DCR B	B AND USE AS A DELAY COUNT			(5)
000:106	302	105	000	JNZ DELAY1	DELAY 15 X FREQ. PARAMETER CLOCK CYCLES			(10)
000:111	033			DCX D	DECREMENT DURATION PARAMETER			(5)
000:112	172			MOV A,D	TEST IF DECREMENTED TO ZERO			(5)
000:113	263			ORA E				(4)
000:114	312	136	000	JZ RETURN	GO RETURN IF SO, CONTINUE IF NOT			(10)
000:117	076	377		MVI A,377H	OUTPUT ONE TO SPEAKER			(7)
000:121	323	XXX		OUT (port address)				(10)
000:123	101			MOV B,C	DELAY AS ABOVE			(5)
000:124	005		DELAY2	DCR B				(5)
000:125	302	124	000	JNZ DELAY2				(10)
000:130	033			DCX D	DECREMENT AND TEST DURATION AS ABOVE			(5)
000:131	172			MOV A,D				(5)
000:132	263			ORA E				(4)
000:133	302	100	000	JNZ TONE	LOOP BACK FOR ANOTHER SQUARE WAVE CYCLE			(10)
000:136	311		RETURN	RET	RETURN TO MAIN PROGRAM			

6:00-6:15 p.m.	0100-0115	Tokyo, Japan	G	15.105
6:00-8:00 p.m.	0100-0300	Moscow, U.S.S.R.	G	12.00, 12.05, 15.18, 17.775 (via Soviet Far East)
		Melbourne, Australia	G	15.32, 17.795
6:00-9:00 p.m.	0100-0400	Madrid, Spain	F	6.065, 11.88 (Mon.-Sat.)
6:30-7:30 p.m.	0130-0230	Tokyo, Japan	G	15.195, 15.39, 17.725, 17.825
7:00-7:15 p.m.	0200-0215	Tokyo, Japan	G	15.105
7:00-7:55 p.m.	0200-0255	Peking, China	G	11.965, 12.055, 15.06
7:00-8:50 p.m.	0200-0350	Taipei, Taiwan	G	11.86, 15.125, 17.72, 17.89
7:30-8:00 p.m.	0230-0300	Stockholm, Sweden	P	9.695, 11.705
8:00-8:15 p.m.	0300-0315	Tokyo, Japan	G	15.105
8:00-8:30 p.m.	0300-0330	Kiev, U.S.S.R.	G	12.05, 15.18, 17.775 (via Soviet Far East)
8:00-8:55 p.m.	0300-0355	Peking, China	G	7.12, 9.78 (via Tirana), 12.055, 15.06 15.385, 17.735, 17.855
8:00-9:25 p.m.	0300-0425	**Johannesburg, S. Africa	F	3.23, 3.995, 4.875
8:00-9:35 p.m.	0300-0435	**San José, Costa Rica	F	6.175, 9.645
8:22-8:28 p.m.	0322-0328	Erevan, U.S.S.R.	G	11.96, 15.13, 15.18, 15.455, (Sat./Tue./Wed./Fri., via Far East)
8:30-9:00 p.m.	0330-0400	Moscow, U.S.S.R.	G	6.02, 9.635, 11.72, 11.96, 12.05, 15.13, 15.18, 15.21, 15.245, (via Soviet Far East)
8:30-9:15 p.m.	0330-0415	Berlin, Ger. Dem. Rep.	P	9.73, 11.84, 11.89
8:30-9:30 p.m.	0330-0430	London, England	G	6.175 (via Sackville), 9.58 (via Ascension)
9:00-9:15 p.m.	0400-0415	Tokyo, Japan	G	15.105
9:00-9:30 p.m.	0400-0430	Sofia, Bulgaria	P	9.70
		Budapest, Hungary	P	6.00, 6.115, 9.585, 9.833, 11.91 (Tue., Fri.)
		Oslo, Norway	F	9.645, 11.87 (Sun.)
9:00-10:00 p.m.	0400-0500	Moscow, U.S.S.R.	G	6.02, 9.635, 11.69, 11.96, 12.05, 15.10, 15.13, 15.18, 15.21, 15.245 (via Far East)
9:00-11:00 p.m.	0400-0600	Montreal, Canada	G	6.135, 9.655
9:15-10:30 p.m.	0415-0530	Bangkok, Thailand	P	9.655, 11.905
9:30-10:00 p.m.	0430-0500	Lisbon, Portugal	P	6.025, 11.935
		Berne, Switzerland	F	9.725, 11.715
		Vienna, Austria	P	6.015
		**London, England	G	6.005, 9.58 (via Ascension)
9:30-10:15 p.m.	0430-0515	**Cologne, Ger. Fed. Rep.	F	7.225 (via Kigali)
10:00-10:15 p.m.	0500-0515	Jerusalem, Israel	F	9.815
		Tokyo, Japan	G	15.105
10:00-10:30 p.m.	0500-0530	Seoul, Rep. of Korea	F	9.64, 15.335
10:00-11:20 p.m.	0500-0620	Hilversum, Holland	G	6.165, 9.715 (via Bonaire)
10:00 p.m.-12 mdt.	0500-0700	H.C.B., Quito, Ecuador	G	6.095, 9.56
		**London, England	G	6.005, 7.27, 9.60 (via Ascension)
10:00 p.m.-12:30 a.m.	0500-0730	Moscow, U.S.S.R.	G	6.02, 9.635, 9.71, 11.69, 11.96, 12.05, 15.10, 15.13, 15.18, 15.21 (via Soviet Far East)
10:30-10:50 p.m.	0530-0550	Cologne, Ger. Fed. Rep.	G	6.10 (via Malta), 6.185, 9.545
11:00-11:15 p.m.	0600-0615	Tokyo, Japan	G	9.505
11:00-11:30 p.m.	0600-0630	Oslo, Norway	P	9.645, 11.87 (Sun.)
11:00-12 mdt.	0600-0700	Buenos Aires, Argentina	G	9.69 (Mon.-Fri.)
11:30 p.m.-1:30 a.m.	0630-0830	Havana, Cuba	G	9.525
11:30 p.m.-1:55 a.m.	0630-0855	**Kuala Lumpur, Malaysia	G	11.90, 15.275
12 mdt.-12:15 a.m.	0700-0715	Tokyo, Japan	G	9.505
12 mdt.-12:30 a.m.	0700-0730	**London, England	G	6.005, 9.60 (via Ascension)
12:30-1:15 a.m.	0730-0815	**London, England	G	9.60, 11.86, 15.40 (via Ascension)
1:00-1:15 a.m.	0800-0815	Tokyo, Japan	G	9.505
2:00-2:15 a.m.	0900-0915	Tokyo, Japan	G	9.505
2:00-2:30 a.m.	0900-0930	Seoul, Rep. of Korea	F	9.64, 15.335
3:00-3:15 a.m.	1000-1015	Tokyo, Japan	G	9.99
3:00-3:50 a.m.	1000-1050	Pyongyang, D.P.R. Korea	G	9.42, 9.535
3:00-4:00 a.m.	1000-1100	**Montreal, Canada	G	5.97, 9.625

*Reception quality, East Coast (West Coast) location: G-good, F-fair, P-poor

**Not intended for North America, but receivable satisfactorily

Frequencies are accurate as of press time, but subject to change. Many stations broadcasting in English but not beamed to North America are often audible here depending upon conditions, locations, time, etc. Sometimes, they are more audible than those nominally beamed here. To help readers and as a sampling of what can be heard, we have added a selection of interesting stations heard here in English. The fine programs from Paris and Brasilia are particularly recommended.



MUSIC SYNTHESIZER KIT

When you combine all of the modules that are part of this package and then throw in keyboard, 12 event sequencer and a four input stereo mixer, it's almost like having two synthesizers in a single package. Wrap them all in sturdy road cases and you have an instrument that goes anywhere and does any job. Module complement includes: Road keyboard with Glide, Two Road module cases, two Balanced Modulator/VCA's, Stereo Mixer, Reverb, three 4720 VCO's, 4730 VCF, two Envelope Generators, three Watt Blocks, Control Oscillator/Noise Source and 12-Event Sequencer.

No. 4700/S Synthesizer Kit \$499.00 *40 lbs.

FREE CATALOG

RAIA ELECTRONICS
DEPT. 9-P

1020 WEST WILSHIRE BLVD.
OKLAHOMA CITY, OK 73116

CIRCLE NO. 49 ON FREE INFORMATION CARD



save on gas!
save on tune-ups!
save on maintenance!

Electronic ignition is "IN"! So says Detroit.

Update your car with either a TIGER CD or a TIGER I breakerless system.

Enjoy the benefits of better gas mileage, quicker starting, elimination of tune-ups, 50,000 miles on points and plugs, and reduced maintenance expenses.

TIGER MAX CD	\$69.95
TIGER 500 CD	59.95
TIGER SST CD	42.95
SIMPLIKIT CD	31.95
TIGER I	45.95

Postpaid U.S.A. only.

Tri-Star Corporation

Dept. ZZ, P.O. Box 1727
Grand Junction, Colorado 81501
CIRCLE NO. 69 ON FREE INFORMATION CARD

a series of numbers could be converted into the series of voltages and thus an audio signal. Waveforms could then be computed directly and generally any possible sound or tone produced. Some very impressive and musically important results have been obtained in this manner. Although this method has perfect generality, the number of computations needed for even a simple piece is staggering. Typically, hours of computer time are required for minutes or even seconds of musical output.

The newest method of musical output is to interface a computer to a sound synthesizer. Units such as a Moog or ARP or some of the circuits described by Don Lancaster in this magazine can be used. The amplitude, frequency, and spectrum of tones produced by the synthesizer are set with control voltages. Contrary to actual sound waveforms, control voltages typically change slowly. This greatly reduces the computational load on the computer. With only \$50 to \$100 worth of components, a multiplexed digital-to-analog converter with a dozen or more control voltage outputs can be built. Ordinary patch cords may then be used to connect the control voltages to the synthesizer.

Timed Loop Techniques. Let's look at the timed-loop technique of Fig. 2. Table of musical notes.

Note	Frequency Hertz	Period Microseconds
C (middle)	261.62	3822.3
C#	277.18	3607.8
D	293.66	3405.3
D#	311.13	3214.2
E	329.63	3033.8
F	349.23	2863.5
F#	369.99	2702.8
G	391.99	2551.1
G#	415.30	2407.9
A	440.00	2272.8
A#	466.16	2145.2
B	493.88	2024.8
C	523.25	1911.2
C#	554.37	1803.9
D	587.33	1702.7
D#	622.25	1607.1
E	659.26	1516.9
F	698.46	1431.8
F#	739.99	1351.4
G	783.99	1275.6
G#	830.61	1204.0
A	880.00	1136.4
A#	932.33	1072.6
B	987.77	1012.4
C	1046.5	955.58

producing single tones with a computer. The two most important elements of music are pitch and rhythm. Curiously, these are the only elements that can be easily controlled in a timed-loop music program. Pitch is determined by the frequency of a tone in cycles per second (Hertz) and rhythm by the relative durations of individual tones in seconds. The computer used must have a definite clock cycle time, and most hobby computers on the market use a crystal clock which gives a very stable and accurate cycle rate. A few of the "trainer" kits may use an RC clock which is not nearly as good, but can still be used to illustrate the principles.

The basic component of a timed-loop music program is the tone-generation subroutine. The tone frequency and duration are passed to the subroutine as two arguments. The main body of the subroutine consists of two "nested" loops. The frequency argument determines how many times the inner loop is executed. The duration argument determines how many times the outer loop is executed. Note that the frequency argument is actually proportional to the period of the wave and the duration is actually the number of cycles of the wave to be played before returning for the next note.

A tone generation subroutine for an 8080 microprocessor is shown in Fig. 1. This routine generates an absolutely symmetrical square wave and uses two inner loops of identical length to accomplish it. To determine what the frequency argument value should be, it is necessary to count up the clock cycles used by the inner wait loop and by the "overhead" instructions outside of the loop. The overhead is found to be 46 cycles and the wait loop is 15 cycles times the value of the frequency parameter. Assuming that the 8080 runs at full speed (0.5 microsecond clock cycle) with no memory waits, the time for a half squarewave cycle is $23 + 7.5N$ microseconds, where N is the frequency parameter. A full cycle time is twice as long. The frequency is, of course, the reciprocal of the full cycle period. Using the routine shown, N must be between 1 and 255; if zero is given, it is interpreted as 256. Thus, the lowest possible frequency is $500,000 / (23 + 7.5 \times 256)$ Hz. or 257.33 Hz. The table in Fig. 2 may be used to determine the proper value of N for any note.

FREE EICO CATALOG

358 Ways To Save On Instruments, Citizens Band, Burglar Alarms, Automotive & Hobby Electronics!

The more you know about electronics, the more you'll appreciate EICO. We have a wide range of products for you to choose from, each designed to provide you with the most pleasure and quality performance for your money. The fact that more than 3 million EICO products are in use attests to their quality and performance.

**"Build-it-Yourself" and save
up to 50% with our famous
electronic kits.**

For latest EICO Catalog and name of nearest EICO Distributor, check reader service card or send 50¢ for fast first class mail service.

**EICO—283 Malta Street,
Brooklyn, N.Y. 11207**

*Leadership in creative electronics
since 1945.*



CIRCLE NO. 23 ON FREE INFORMATION CARD

The duration parameter is actually a count of half cycles to be played before returning. As a result, the argument value is a function of both the duration in seconds and the note played. Thus $M=2TF$, where M is the argument value, T is the time in seconds, F is the frequency in Hertz, and the factor of 2 accounts for both halves of the square wave. Note that the routine of Fig. 1 uses a double precision duration argument to provide for long notes.

Fig. 3. Basic specification for music language NOTRAN.

1. TEMPO statement

Example: TEMPO $\frac{1}{4}=500$

TEMPO is keyword identifier.

$\frac{1}{4}$ refers to a "quarter note."

500 is duration of quarter note in milliseconds.

2. Note statement

Example: 1C#4, $\frac{1}{8}$

1 refers to voice 1 (optional).

C is note name, # following is sharp, @ is flat.

4 is octave number, C4 is middle C.

$\frac{1}{8}$ specifies an eighth note.

3. Rest statement

Example: R, $\frac{1}{2}$

R signifies rest

$\frac{1}{2}$ means rest duration equivalent of half note.

4. For monophonic music, one statement per line

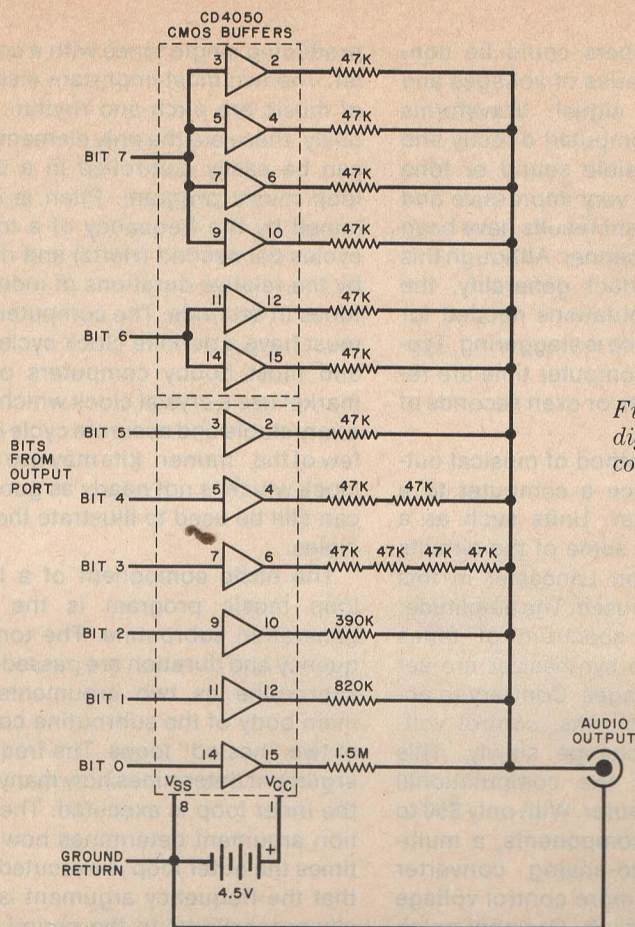
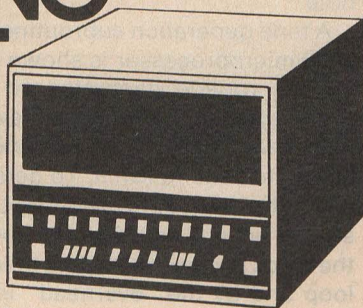


Fig. 4. Simple 8-bit digital-to-analog converter.

THINKING ABOUT YOUR OWN COMPUTER?



Join over 50,000 avid readers of BYTE, the magazine with rich, professionally edited articles on micro-computers . . . for building, expanding and having downright fun with your own system. You'll reread super articles on . . .

- detailed hardware/software designs by successful experimenters and hobbyists
- reviews of upcoming general purpose systems
- editorials on the fun of computers . . . electronic music, video games, hobbyist control systems, ideas for ham radio, model railroading and lots more
- tutorial background and sources full of ideas for home computers and computer science
- ads by firms with computer products you want
- club information and social activities

SUBSCRIBE TO BYTE NOW! IT'S FUN. . .AND GLITCH-PROOF!

Send this coupon for a trial subscription to BYTE. Get your first issue by return mail. Read it from cover-to-cover. If it isn't everything you want, just write "CANCEL" on the bill and return it to us. The first copy is yours to keep.

BYTE PETERBOROUGH, NH 03458

Please enter my trial subscription to BYTE . . .

☐ \$12 One Year ☐ \$22 Two Years ☐ \$30 Three Years

I understand you will send the first issue by return mail and bill me later. If I don't like BYTE, I just write "CANCEL" across the invoice and return it. I will not be charged.

Name (Please Print) _____

Address _____

City _____ State _____ Zip _____



Using the subroutine in a music program is simply a matter of calling it with the right frequency and duration arguments for each note to be played. Care must be taken to insert some silence between notes of the same frequency or else they will run together. A simple-minded program might fetch these from a hand-prepared list stored in memory. A more challenging approach is to write a program that accepts statements written in a simple "music language" and computes the tone-generation arguments (Fig. 3). Then your musically inclined wife, for example, may perpare scores for the computer herself.

Improving Basic Timed Loop Techniques.

For better sounding music, a number of refinements can be made to the basic technique. It is possible to get some limited change of tone color by changing the duty cycle of the wave. This may be accomplished by inserting some "no operation" instructions in one of the wait loops. Unfortunately, the frequency arguments will have to be recomputed. Good duty cycle values to try are $\frac{1}{3}$ and $\frac{1}{4}$, since harmonics divisible by 3 and 4, respectively, will be missing from the spectra.

Another interesting experiment is to improve the frequency accuracy of the routine. As written, the half-cycle time is in increments of 7.5 microseconds. This results in significant frequency errors, particularly for high notes. By inserting additional selected overhead instructions, it is possible to be as accurate as 0.5 microsecond. This might be accomplished by writing 15 copies of the routine with extra instructions added in increments of one clock cycle. Obtaining the right note would then be a combination of calling the proper routine with the proper frequency argument.

The computer-generated tones can be made more natural sounding if an amplitude envelope is added to the notes. All this amounts to is changing the volume in a controlled manner over the duration of the note. Additionally, there would then be some control over the dynamics (changes in loudness during a piece) of the music. At this point, some specialized hardware must be added.

The circuit in Fig. 4 may be added to an output port. This is actually a very simple 8-bit digital-to-analog converter. Best results are obtained if the 47k

resistors are 5% carbon film types from the same batch. Several mail-order firms now offer this type of resistor. A battery is used to power the circuit so that system noise will be isolated. Any type of audio amplifier can be connected to the circuit's output.

The dc output voltage of this circuit is 0.0176N volts, where N is the binary number last sent to the output port. Note, in the tone generation subroutine, that 0 and 255 (decimal) were alternately sent to the output port for a square wave. If this circuit was connected to the output port, the output voltage would alternately switch between 0 volts and +4.5 volts. If instead, 0 and 128 were alternately sent out, then the wave would switch between 0 volts and about 2.26 volts. This is only about one-half of the previous amplitude so the tone would not be as loud. Thus, amplitude control may be exercised by simply changing the values sent to the output port by the tone-generation subroutine.

Unfortunately, programming amplitude envelopes during a tone without upsetting the precise loop timing is a bit tricky. As the level of sophistication increases, a point is reached where

the effort needed to preserve the loop timing is excessive. At this point specialized hardware is needed to relieve the computer of the actual tone generation task.

Experiments with Other Sounds.

Other sounds can also be easily generated by a computer. For example, a high-speed stream of random bits will provide a good source of white noise. An 8080 program to generate white noise is given in Fig. 5. Random bits are generated by simulating a 16-stage shift register with appropriate feedback. White noise might be used in a rhythm program to approximate the sound of a snare drum.

The program in Fig. 6 produces some really "way-out" sounds. One should first try it and listen to each of the 16 possible output bits. Then study the program and see if you can figure out what it is actually doing. Finally, try to explain the weird results near the end of each repetition.

Many other interesting experiments can be performed with the simple digital-to-analog converter described earlier. Some of these will be described next time. ♦

Fig. 5. White noise generation routine for 8080

```

000:000 021 000 000 WHITEN LXI D,0          SET D & E TO ZERO
000:003 041 001 000        LXI H,1          RET SHIFT REG. IN H & L NON-ZERO
000:006 174          LOOP MOV A,H          OUTPUT UPPER 8 BITS OF SHIFT REGISTER
000:007 323 XXX        OUT (port address) AS A SET OF RANDOM BITS
000:011 017          RRC                   FORM EXCLUSIVE-OR OF SHIFT REGISTER
000:012 254          XRA H                BITS 15,14,12, AND 3 IN BIT 0 OF A
000:013 017          RRC
000:014 017          RRC
000:015 254          XRA H
000:016 017          RRC
000:017 255          XRA L
000:020 017          RRC
000:021 017          RRC
000:022 017          RRC
000:023 346 001        ANI 1              ISOLATE BIT 0 OF A
000:025 051          DAD H                SHIFT SHIFT REGISTER LEFT 1
000:026 137          MOV E,A             AND BRING BIT 0 OF A INTO
000:027 031          DAD D                VACATED SHIFT REGISTER BIT 0
000:030 303 006 000    JMP LOOP          LOOP

```

Fig. 6. Weird sound generation routine for the 8080.

* CONNECT SPEAKER TO ANY BIT OF OUTPUT PORT
* A DIFFERENT SOUND WILL BE HEARD AT EACH BIT

```

000:000 021 001 000 WEIRD LXI D,1          INITIALIZE
000:003 041 000 000        LXI H,0
000:006 175          LOOP MOV A,L          CHANGE TO: MOV A,H FOR 8 MORE SOUNDS
000:007 323 XXX        OUT (port address) OUTPUT TO SPEAKER
000:011 031          DAD D
000:012 322 006 000    JNC LOOP
000:015 023          INX D
000:016 172          MOV A,D
000:017 263          ORA E
000:020 302 006 000    JNZ LOOP
000:023 023          INX D
000:024 303 006 000    JMP LOOP

```




Electronics Library

TRANSISTOR BASICS: A SHORT COURSE (REVISED SECOND EDITION)

by George Stanley, Jr.

The revised edition of this practical guide covers recent transistors and circuit appli-

cations. Beginning with an introduction to commonly used terms, the author describes diode and transistor behavior. Circuit analysis is conducted on a practical level so that the reader will be able to determine required circuit parameters for many applications. Amplifier action, feedback, and special solid-state devices are explored. Troubleshooting information for bipolar and field-effect transistor circuits is included.

Published by Hayden Book Co., 50 Essex St., Rochelle Park, NJ 07662. 128 pages. \$4.30 soft cover.

RCA RECEIVING TUBE MANUAL

The new edition of the RCA Receiving Tube Manual, RC-30, contains chapters on basic

operational principles of vacuum tubes, electrical characteristics, and circuit applications. Technical data are included on active as well as discontinued tube types. Among the book's features are tube basing diagrams, a picture tube characteristics chart, and an expanded replacement guide for entertainment and industrial receiving tubes.

Published by RCA Distributor and Special Products Division, Box 85, Runnemede, NJ 08078. 754 pages, soft cover. \$3.45 post-paid.

TRANSISTOR THEORY FOR TECHNICIANS AND ENGINEERS

by Andrew Veronis

Semiconductor parameters, device design and fabrication, transistor operation, and circuit theory are explored in this book. The author begins with a description of atomic structure, then progresses to electron action within doped semiconductors and pn junctions. Diode and transistor behavior is explained in terms of important properties — reverse saturation current, diode resistance, current transfer, emitter injection efficiency, avalanche multiplication, etc. Transistor parameters are discussed in relation to circuit performance. Analyses of basic transistor circuits, such as common emitter, common base, and common collector, are given.

Published by Tab Books, Blue Ridge Summit, PA 17214. 224 pages. \$8.95 hard cover, \$5.95 soft cover.

SOLID-STATE CIRCUIT ANALYSIS

by Vester Robinson

Solid-state devices are analyzed from a practical circuit applications point of view. Simple pn diodes, bipolar transistors, tunnel and zener diodes, field effect transistors, unijunction transistors, and integrated circuits are examined in detail. Sample circuits include many commonly found in AM and FM transmitters and receivers. Mathematics is kept to a low level (mostly algebra) and is used only when necessary. Review questions appear at the end of each chapter.

Published by Reston Publishing Co., Box 547, Reston, VA 22090. 372 pages. \$15.95 hard cover.

ACTIVE FILTER COOKBOOK

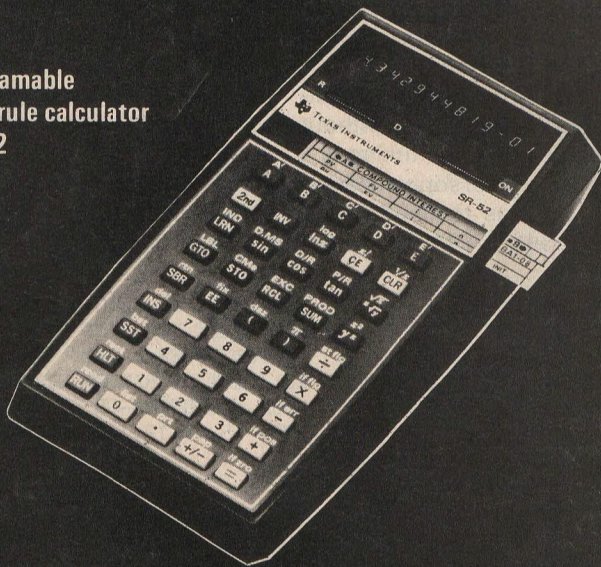
by Don Lancaster

The compact, high-Q filters made possible by op-amp technology are described in this book. Among those covered are low- and high-pass filters, bandpass filters, and switchable and voltage-controlled designs. Thorough design information as well as ready-to-use circuits are presented. Operating ranges vary from subsonics to ultrasonics.

Published by Howard W. Sams & Co., 4300 W. 62nd St., Indianapolis, IN 46206. 192 pages. \$5.95 soft cover.

NOW FROM TEXAS INSTRUMENTS . . . three machines in one.

programmable
slide-rule calculator
SR-52



- 10 user defined keys
- 224 program storage locations
- 23 preprogrammed key functions
- 8 preprogrammed condition statements
- 20 independent addressable memory registers
- Permanent program storage on magnetic cards

It took TEXAS INSTRUMENTS to invent the SR-52 calculator. It took C & S MARKETING ASSOCIATES to offer it at a price you can afford, now only \$249.95. With such versatility and such an affordable price, you can not afford to be without the problem solving power of card programmability. Now solve problems in seconds that would take hours with an ordinary calculator or sliderule if they could be done at all.

For more information or the answer to any question you may have about the SR-52 calculator, call toll free (800-251-6771)*. Tenn. residents call (800-252-6706). Other TEXAS INSTRUMENT models available from \$49.95.

Each TEXAS INSTRUMENT calculator comes with a 1-year warranty. Should your unit prove defective within 60 days, just return it for a new unit! Finally should you be dissatisfied with your calculator return it within 15 days for a prompt refund.

C & S MARKETING ASSOC. P.O. BOX 165 ALGOOD, TENN. 38501

QTY. _____ PRICE 249.95 ea.

☐ CHECK ☐ M.O. ☐ C.O.D.

☐ LITERATURE ☐ INFORMATION

DIAL (800-251-6771) IN TENN. DIAL (800-252-6706).

BYTE SHOP OF WESTMINSTER DISASSEMBLER - DIS
VERSION 1.5
COPYRIGHT (C) 1977, 1978

SYMBOL TABLE CLEARED

0000	31 82 0D	1	0000	LXI	SP, 0D82	✓
0003	CD AC 09			CALL	09AC CROUT	✓
0006	21 6D 0B	!m		LXI	H, 0B6D	✓
0009	CD DA 01			CALL	01DA	✓
000C	21 93 0B	!		LXI	H, 0B93	✓
000F	CD DA 01			CALL	01DA	✓
0012	21 B9 0B	!		LXI	H, 0BB9	✓
0015	CD DA 01			CALL	01DA	✓
0018	CD AC 09			CALL	09AC CROUT	✓
001B	CD AF 05			CALL	05AF	✓
001E	21 95 0A	!		LXI	H, 0A95	✓
0021	22 7D 0A	"3		SHLD	0A7D	✓
0024	21 CB 0A	!		LXI	H, 0ACB	✓
0027	22 7F 0A	"		SHLD	0A7F	✓
002A	21 01 0B	!		LXI	H, 0B01	✓
002D	22 79 0A	"u		SHLD	0A79	✓
0030	21 37 0B	!7		LXI	H, 0B37	✓
0033	22 7B 0A	"c		SHLD	0A7B	✓
0036	AF			XRA	A	✓
0037	32 74 0A	2t		STA	0A74	✓
003A	32 78 0A	2x		STA	0A78	✓
003D	32 82 0A	2		STA	0A82	✓
0040	32 81 0A	2		STA	0A81	✓
0043	32 75 0A	2u		STA	0A75	✓
0046	32 76 0A	2v		STA	0A76	✓
0049	3E 10	>		MVI	A, 10	✓
004B	32 77 0A	2w		STA	0A77	✓
004E	3E EE	>		MVI	A, EE	✓
0050	32 4D 0A	2M		STA	0A4D	✓
0053	32 4E 0A	2N		STA	0A4E	✓
0056	21 F5 0B	!		LXI	H, 0BF5	✓
0059	CD 5C 01	\		CALL	015C STRG	✓
005C	CD C9 09			CALL	09C9 WHD	✓
005F	CD BF 09			CALL	09BF WHJ	✓
0062	78	x		<u>MOV</u>	<u>A, B</u> NOP	✓
0063	0F			RRC		✓
0064	D2 6F 00	0		JNC	006F	✓
0067	3E 01	>		MVI	A, 01	✓
0069	32 82 0A	2		STA	0A82	✓
006C	CD C3 05			CALL	05C3	✓
006F	CD 42 02	B		CALL	0242	✓
0072	CD AC 09			CALL	09AC CROUT	✓
0075	31 82 0D	1		LXI	SP, 0D82	✓
0078	21 E6 0D	!		LXI	H, 0DE6	✓
007B	22 59 0A	"Y		SHLD	0A59	✓
007E	21 20 20	!		LXI	H, 2020	✓
0081	22 20 0C	"		SHLD	0C20	✓
0084	21 10 0C	!		LXI	H, 0C10	✓
0087	CD 5C 01	\		CALL	015C STRG	✓
008A	CD 20 03			CALL	0320	✓
008D	CD AC 09			CALL	09AC CROUT	✓
0090	21 0A 0D	!		LXI	H, 0D0A	✓
0093	7E	~		MOV	A, M	✓
0094	FE 47	0		CPI	47	✓
0096	CA D7 00			JZ	00D7	✓
0099	FE 44	D		CPI	44	✓
009B	CA 42 03	B		JZ	0342	✓
009E	FE 45	E		CPI	45	✓
00A0	CA 2C 02	,		JZ	022C	✓
00A3	FE 53	S		CPI	53	✓
00A5	CA E1 01			JZ	01E1	✓
00A8	FE 4F	0		CPI	4F	✓

H0018 -

* DO YOU WANT WHITE (Y, N) *

{ if 0 of Y = 1
" " " N = 0

* because
Jump if 'No'

H006F
H0072
H0075
H0078 -

* : 0000 *

* G' - Go?

* D' - Display Board?

* E' - Exchange Sides?

* S' - Set Mode? 0000 000000

* O' *

00AF	CA 20 02	
00B2	FE 41	A
00B4	CA F1 02	
00B7	FE 4E	N
00B9	CA 05 03	
00BC	CD 68 01	h
00BF	3A 0F 0D	!
00C2	FE 4D	M
00C4	CA 26 01	&
00C7	FE 3D	=
00C9	CA 48 03	H
00CC	FE 0D	
00CE	C2 2C 01	,
00D1	CD 6D 08	m
00D4	CD 42 02	B
00D7	CD 7D 04	3
00DA	3A 78 0A	!x
00DD	B7	
00DE	CA F4 00	
00E1	21 4F 20	10
00E4	22 1E 0C	"
00E7	21 20 4F	! 0
00EA	22 1B 0C	"
00ED	AF	
00EE	32 78 0A	2x
00F1	C3 02 01	
00F4	CD AD 01	
00F7	CD 97 03	
00FA	3A 4F 0A	!0
00FD	FE FF	
00FF	CA 38 01	8
0102	21 16 0C	!
0105	CD 5C 01	\
0108	3A 81 0A	!
010B	FE FF	
010D	CA 17 03	
0110	3A 74 0A	!t
0113	B7	
0114	CA 1D 01	
0117	21 AE 0C	!
011A	CD 5C 01	\
011D	CD AC 09	
0120	CD 42 02	B
0123	C3 75 00	u
0126	CD 6D 08	m
0129	C3 75 00	u
012C	21 23 0C	!#
012F	CD 5C 01	\
0132	CD AC 09	
0135	C3 75 00	u
0138	21 30 0C	!0
013B	CD DA 01	
013E	CD 42 02	B
0141	CD AC 09	
0144	21 47 0C	!G
0147	CD 5C 01	\
014A	CD C9 09	
014D	CD BF 09	
0150	78	x
0151	FE 59	Y
0153	C2 C8 01	
0154	CD AC 09	
0159	C3 18 00	
015C	78	
015F	FE 0D	

JZ	0220	✓	'R' — Design
CPI	41	✓	'A'
JZ	02F1	✓	'N'
CPI	4E	✓	'N'
JZ	0305	✓	
CALL	0168	✓	
LDA	0D0F	✓	'm' scribble
CPI	4D	✓	'm'
JZ	0126	✓	'='
CPI	3D	✓	'='
JZ	0348	✓	
CPI	0D	✓	cr
JNZ	012C	✓	
CALL	086D	✓	
CALL	0242	✓	
CALL	047D	✓	Command
LDA	0A78	✓	
ORA	A	✓	
JZ	00F4	✓	
LXI	H, 204F	✓	
SHLD	0C1E	✓	
LXI	H, 4F20	✓	
SHLD	0C1B	✓	
XRA	A	✓	
STA	0A78	✓	
JMP	0102	✓	
CALL	01AD	✓	0,0,0
CALL	0397	✓	0,0,0
LDA	0A4F	✓	
CPI	FF	✓	
JZ	0138	✓	
LXI	H, 0C16	✓	'MC: 74-34'
CALL	015C	✓	STRG
LDA	0A81	✓	
CPI	FF	✓	
JZ	0317	✓	
LDA	0A74	✓	
ORA	A	✓	
JZ	011D	✓	'CHECK!'
LXI	H, 0CAE	✓	
CALL	015C	✓	STRG
CALL	09AC	✓	CROUT
CALL	0242	✓	
JMP	0075	✓	
CALL	086D	✓	
JMP	0075	✓	
LXI	H, 0C23	✓	INPUT ERROR
CALL	015C	✓	STRG
CALL	09AC	✓	CROUT
JMP	0075	✓	
LXI	H, 0C30	✓	'CHECKMATE 100 WIN!!'
CALL	01DA	✓	
CALL	0242	✓	
CALL	09AC	✓	CROUT
LXI	H, 0C47	✓	PLAY AGAIN? (Y,N)
CALL	015C	✓	STRG
CALL	09C7	✓	WHP
CALL	09BF	✓	WHJ
MOV	A, B	✓	
CPI	59	✓	'Y'
JNZ	01C8	✓	
CALL	09AC	✓	CROUT
JMP	0018	✓	
MOV	A, M	✓	

STRG

0160	47	G	MOV	B,A	✓
0161	CD BF 09		CALL	09BF	✓ WH1
0164	23	#	INX	H	✓
0165	C3 5C 01	\	JMP	015C	✓
0168	2A 0A 0D	* H0168	LHLD	0D0A	✓
016B	CD 99 01		CALL	0199	✓
016E	32 50 0A	2P	STA	0A50	✓
0171	2A 0D 0D	*	LHLD	0D0D	✓
0174	CD 99 01		CALL	0199	✓
0177	32 51 0A	2Q	STA	0A51	✓
017A	32 4E 0A	2N	STA	0A4E	✓
017D	3A 50 0A	!P	LDA	0A50	✓
0180	21 0C 0A	!	LXI	H,0A0C	✓
0183	0E 1F		MVI	C,1F	✓
0185	BE	H0185	CMP	M	✓
0186	CA 91 01		JZ	0191	✓
0189	2B	+	DCX	H	✓
018A	0D		DCR	C	✓
018B	F2 85 01		JP	0185	✓
018E	C3 2C 01	,	JMP	012C	✓
0191	79	H H0191	MOV	A,C	✓
0192	32 4D 0A	2M	STA	0A4D	✓
0195	32 4F 0A	2D	STA	0A4F	✓
0198	C9		RET		✓
0199	7D	J H0199	MOV	A,L	✓
019A	E6 0F		ANI	0F	✓
019C	17		RAL		✓
019D	17		RAL		✓
019E	17		RAL		✓
019F	17		RAL		✓
01A0	47	G	MOV	B,A	✓
01A1	7C	!	MOV	A,H	✓
01A2	E6 0F		ANI	0F	✓
01A4	B0		ORA	B	✓
01A5	47	G	MOV	B,A	✓
01A6	E6 88		ANI	88	✓
01A8	C2 2C 01	,	JNZ	012C	✓
01AB	78	x	MOV	A,B	✓
01AC	C9		RET		✓
01AD	3A 50 0A	!P	LDA	0A50	
01B0	47	G	MOV	B,A	
01B1	CD 8F 09		CALL	098F	
01B4	2A DD 09	*	LHLD	09DD	
01B7	22 1B 0C	"	SHLD	0C1B	
01BA	3A 51 0A	!Q	LDA	0A51	
01BD	47	G	MOV	B,A	
01BE	CD 8F 09		CALL	098F	
01C1	2A DD 09	*	LHLD	09DD	
01C4	22 1E 0C	"	SHLD	0C1E	
01C7	C9		RET		
01C8	CD AC 09		CALL	09AC	CROUT ✓
01CB	CD AC 09		CALL	09AC	CROUT ✓
01CE	21 5B 0C	!C	LXI	H,0C5B	
01D1	CD 5C 01	\	CALL	015C	STRG ✓
01D4	CD AC 09		CALL	09AC	CROUT ✓
01D7	C3 01 C0		JMP	C001	go to monitor. ✓
01DA	CD 5C 01	\ H01DA	CALL	015C	STRG ✓
01DD	CD AC 09		CALL	09AC	CROUT ✓
01E0	C9		RET		✓
01E1	CD AC 09	H01E1	CALL	09AC	CROUT ✓
01E4	21 DF 0B	!	LXI	H,0BDF	WHICH mode ✓
01E7	CD 5C 01	\	CALL	015C	STRG ✓
01EA	CD C9 09		CALL	09C9	WH1 ✓
01ED	CD BF 09		CALL	09BF	WH1 ✓

end of game
 }
 go to monitor.
 S command
 MODE RTN

01F3	CA 03 02		JZ	0203	✓ 'S'
01F6	FE 42	B	CPI	42	✓ 'B'
01F8	CA 0A 02		JZ	020A	✓
01FB	FE 4E	N	CPI	4E	✓ 'N'
01FD	CA 11 02		JZ	0211	✓
0200	C3 2C 01	,	JMP	012C	✓
0203	06 00		MVI	B,00	✓
0205	0E FF		MVI	C,FF	✓
0207	C3 15 02		JMP	0215	✓
020A	06 00		MVI	B,00	✓
020C	0E FB		MVI	C,FB	✓
020E	C3 15 02		JMP	0215	✓
0211	06 08		MVI	B,08	✓
0213	0E FB		MVI	C,FB	✓
0215	78	X	MOV	A,B	✓
0216	32 1E 07	2	STA	071E	✓
0219	79	W	MOV	A,C	✓
021A	32 61 08	2B	STA	0861	✓
021D	C3 72 00	r	JMP	0072	✓
0220	CD AC 09		CALL	09AC	✓
0223	21 7E 0C	1~	LXI	H,0C7E	✓
0226	CD 5C 01	\	CALL	015C	✓
0229	C3 3E 01	>	JMP	013E	✓
022C	CD C3 05		CALL	05C3	✓
022F	3A 82 0A	:	LDA	0A82	✓
0232	B7		ORA	A	✓
0233	CA 3A 02	:	JZ	023A	✓
0236	AF		XRA	A	✓
0237	C3 3C 02	<	JMP	023C	✓
023A	3E 01	>	MVI	A,01	✓
023C	32 82 0A	2	STA	0A82	✓
023F	C3 75 00	u	JMP	0075	✓
0242	CD AC 09		CALL	09AC	✓
0245	CD AC 09		CALL	09AC	✓
0248	16 00		MVI	D,00	✓
024A	21 B6 0C	!	LXI	H,0CB6	✓
024D	CD DA 01		CALL	01DA	✓
0250	7A	Z	MOV	A,D	✓
0251	B7		ORA	A	✓
0252	CA 5B 02	E	JZ	025B	✓
0255	21 D2 0C	!	LXI	H,0CD2	✓
0258	00		NOP		✓
0259	00		NOP		✓
025A	00		NOP		✓
025B	06 21	!	MVI	B,21	✓
025D	CD BF 09		CALL	09BF	✓
0260	06 20		MVI	B,20	✓
0262	CD BF 09		CALL	09BF	✓
0265	21 0C 0A	!	LXI	H,0A0C	✓
0268	0E 1F		MVI	C,1F	✓
026A	7E	~	MOV	A,M	✓
026B	BA		CMP	D	✓
026C	CA B8 02		JZ	02B8	✓
026F	2B	+	DCX	H	✓
0270	0D		DCR	C	✓
0271	F2 6A 02	J	JP	026A	✓
0274	7A	Z	MOV	A,D	✓
0275	E6 0F		ANI	0F	✓
0277	5F		MOV	E,A	✓
0278	7A	Z	MOV	A,D	✓
0279	E6 F0		ANI	F0	✓
027B	0F		RRC		✓
027C	0F		RRC		✓
027D	0F		RRC		✓
027E	0F		RRC		✓

LXI B, 00FF } SuperBlitz

LXI B, 00FB } Blitz

LXI B, 00FB } Noire

Resign (R) command

YOU RESIGNED - I WIN!

E command

! --- MICROCHESS --- !

0280	1F		RAR		✓
0281	DA 89 02		JC	0289	✓
0284	06 20		MVI	B, 20	✓
0286	C3 8B 02		JMP	028B	✓
0289	06 3A	!	MVI	B, 3A	✓
028B	CD BF 09		CALL	028F	WH1 ✓
028E	CD BF 09		CALL	028F	WH1 ✓
0291	06 20		MVI	B, 20	✓
0293	CD BF 09		CALL	028F	WH1 ✓
0296	14		INR	D	✓
0297	7A	Z	MOV	A, D	✓
0298	E6 0F		ANI	0F	✓
029A	FE 08		CPI	08	✓
029C	C2 65 02	e	JNZ	0265	✓
029F	06 21	!	MVI	B, 21	✓
02A1	CD BF 09		CALL	028F	WH1 ✓
02A4	CD AC 09		CALL	02AC	CROUT ✓
02A7	7A	Z	MOV	A, D	✓
02A8	C6 08		ADI	08	✓
02AA	57	W	MOV	D, A	✓
02AB	F2 50 02	P	JP	0250	✓
02AE	21 EE 0C	!	LXI	H, 0CEE	✓
02B1	CD DA 01		CALL	01DA	✓
02B4	CD AC 09		CALL	02AC	CROUT ✓
02B7	C9		RET		✓
02B8	79	W	MOV	A, C	✓
02B9	FE 10		CPI	10	✓
02BB	D2 D3 02		JNC	02D3	✓
02BE	3A 82 0A	!	LDA	0A82	✓
02C1	B7		ORA	A	✓
02C2	C2 CC 02		JNZ	02CC	✓
02C5	3A 83 0A	!	LDA	0A83	✓
02C8	47	B	MOV	B, A	✓ } W
02C9	C3 DD 02		JMP	02DD	✓
02CC	3A 84 0A	!	LDA	0A84	✓ } B
02CF	47	B	MOV	B, A	✓ } B
02D0	C3 DD 02		JMP	02DD	✓
02D3	3A 82 0A	!	LDA	0A82	✓ } print A or B
02D6	B7		ORA	A	✓
02D7	C2 C5 02		JNZ	02C5	✓
02DA	C3 CC 02		JMP	02CC	✓
02DD	CD BF 09		CALL	028F	WH1 ✓
02E0	79	W	MOV	A, C	✓
02E1	E6 0F		ANI	0F	✓
02E3	4F	0	MOV	C, A	✓
02E4	06 00		MVI	B, 00	✓
02E6	21 85 0A	!	LXI	H, 0A85	✓
02E9	09		DAD	B	✓
02EA	46	F	MOV	B, M	✓
02EB	CD BF 09		CALL	028F	WH1 ✓
02EE	C3 91 02		JMP	0291	✓
02F1	3E CD	>	MVI	A, CD	✓
02F3	32 D4 00	2	STA	00D4	✓
02F6	32 20 01	2	STA	0120	✓
02F9	21 42 02	!B	LXI	H, 0242	✓
02FC	22 D5 00	"	SHLD	00D5	✓
02FF	22 21 01	"!	SHLD	0121	✓
0302	C3 72 00	"	JMP	0072	✓
0305	21 00 00	!	LXI	H, 0000	✓
0308	22 D4 00	"	SHLD	00D4	✓
030B	22 D5 00	"	SHLD	00D5	✓
030E	22 20 01	"	SHLD	0120	✓
0311	22 21 01	"!	SHLD	0121	✓
0314	C3 72 00	"	JMP	0072	✓

H0289 -
H028B -
H0291 -

H02B8

H02C5

H02CC

H02D3

H02DD

H02F1

H0305

H0317

print empty squares

! CHALLENGER !

BLACK IF H0A282=0
WHITE OTHERWISE

print pieces

A command

call 242

N Command

CHECKMATE I WIN!

031A	CD 5C 01	>	CALL	015C	CALL STRG	
031D	C3 3E 01	>	JMP	013E		
0320	21 0A 0D	I	LXI	H, 0D0A	✓	Line Input Handler
0323	0E 00		MVI	C, 00	✓	
0325	CD C9 09		CALL	07C9	CALL WHØ ✓	
0328	78	x	MOV	A, B	✓	
0329	77	w	MOV	M, A	✓	
032A	FE 0D		CPI	0D	VCR	
032C	CA 3C 03	<	JZ	033C	✓	
032F	FE 7F		CPI	7F	✓ RUBOUT (RUSPC)	
0331	CA 30 0E	0	JZ	0E30	✓ RUBOUT rtn	
0334	CD BF 09		CALL	09BF	WHI ✓	
0337	23	#	INX	H	✓	
0338	0C		INR	C	✓	
0339	C3 25 03	%	JMP	0325	✓	
033C	AF		XRA	A	✓	
033D	B9		CMP	C	✓	
033E	CA 25 03	%	JZ	0325	✓	
0341	C9		RET		✓	
0342	CD 42 02	B	CALL	0242	✓	D command (Display?)
0345	C3 75 00	u	JMP	0075	✓	
0348	CD 68 01	h	CALL	0168	✓	= command
034B	21 0C 0A	I	LXI	H, 0A0C	✓	
034E	0E 0F		MVI	C, 0F	✓	
0350	3A 50 0A	:P	LDA	0A50	✓	
0353	BE		CMP	M	✓	
0354	CA 5C 03	\	JZ	035C	✓	
0357	2B	+	DCX	H	✓	
0358	0D		DCR	C	✓	
0359	F2 53 03	S	JP	0353	✓	
035C	36 CC	6	MVI	M, CC	✓	
035E	21 8C 0A	I	LXI	H, 0A8C	✓	
0361	0E 07		MVI	C, 07	✓	
0363	3A 10 0D	:	LDA	0D10	✓	
0366	BE		CMP	M	✓	
0367	CA 72 03	r	JZ	0372	✓	
036A	2B	+	DCX	H	✓	
036B	0D		DCR	C	✓	
036C	F2 66 03	f	JP	0366	✓	
036F	C3 2C 01	,	JMP	012C	✓	
0372	21 FD 09	I	LXI	H, 09FD	✓	
0375	06 00		MVI	B, 00	✓	
0377	09		DAD	B	✓	
0378	3E CC	>	MVI	A, CC	✓	
037A	BE		CMP	M	✓	
037B	CA 83 03		JZ	0383	✓	
037E	2B	+	DCX	H	✓	
037F	BE		CMP	M	✓	
0380	C2 2C 01	,	JNZ	012C	✓	
0383	3A 51 0A	:Q	LDA	0A51	✓	
0386	77	w	MOV	M, A	✓	
0387	3A 11 0D	t	LDA	0D11	✓	
038A	FE 4D	M	CPI	4D	✓	
038C	CA 72 00	r	JZ	0072	✓	
038F	FE 0D		CPI	0D	✓	
0391	CA D4 00		JZ	00D4	✓	
0394	C3 2C 01	,	JMP	012C	✓	
0397	3A 51 0A	:Q	LDA	0A51	✓	
039A	E6 F0		ANI	F0	✓	
039C	FE 70	P	CPI	70	✓	
039E	C0		RNZ		✓	
039F	3A 4F 0A	:D	LDA	0A4F	✓	
03A2	4F	0	MOV	C, A	✓	
03A3	E6 08		ANI	08	✓	
03A5	CB		RZ		✓	
03A6	21 FD 09	I	LXI	H, 09FD	✓	

03A7	00 00		MVI	B,00
03AB	09		DAD	B
03AC	3E CC	>	MVI	A,CC
03AE	77	W	MOV	M,A
03AF	21 EE 09	I	LXI	H,09EE
03B2	1E 00		MVI	E,00
03B4	BE		CMP	M
03B5	CA BD 03		JZ	03BD
03B8	23	#	INX	H
03B9	1C		INR	E
03BA	C3 B4 03		JMP	03B4
03BD	3A 51 0A	!Q	LDA	0A51
03C0	77	W	MOV	M,A
03C1	3E 3D	>=	MVI	A,3D
03C3	32 20 0C	2	STA	0C20
03C6	21 86 0A	I	LXI	H,0A86
03C9	16 00		MVI	D,00
03CB	19		DAD	D
03CC	7E	~	MOV	A,M
03CD	32 21 0C	2I	STA	0C21
03D0	C9		RET	
03D1	3A 0C 0D	!	LDA	0D0C
03D4	FE 4F	0	CPI	4F
03D6	C2 2C 01	,	JNZ	012C
03D9	3A 0E 0D	!	LDA	0D0E
03DC	FE 4F	0	CPI	4F
03DE	C2 2F 04	/	JNZ	042F
03E1	3A 82 0A	!	LDA	0A82
03E4	B7		ORA	A
03E5	CA 05 04		JZ	0405
03E8	3A FD 09	!	LDA	09FD
03EB	FE 74	t	CPI	74
03ED	C2 2C 01	,	JNZ	012C
03F0	3E 72	>r	MVI	A,72
03F2	32 FD 09	2	STA	09FD
03F5	3E 73	>s	MVI	A,73
03F7	32 00 0A	2	STA	0A00
03FA	32 4E 0A	2N	STA	0A4E
03FD	3E 13	>	MVI	A,13
03FF	32 4D 0A	2M	STA	0A4D
0402	C3 1F 04		JMP	041F
0405	3A FD 09	!	LDA	09FD
0408	FE 73	s	CPI	73
040A	C2 2C 01	,	JNZ	012C
040D	3E 75	>u	MVI	A,75
040F	32 FD 09	2	STA	09FD
0412	3E 74	>t	MVI	A,74
0414	32 00 0A	2	STA	0A00
0417	32 4E 0A	2N	STA	0A4E
041A	3E 13	>	MVI	A,13
041C	32 4D 0A	2M	STA	0A4D
041F	3A 0F 0D	!	LDA	0D0F
0422	FE 4D	M	CPI	4D
0424	CA 72 00	r	JZ	0072
0427	FE 0D		CPI	0D
0429	CA D4 00		JZ	00D4
042C	C3 2C 01	,	JMP	012C
042F	3A 82 0A	!	LDA	0A82
0432	B7		ORA	A
0433	CA 53 04	s	JZ	0453
0436	3A FD 09	!	LDA	09FD
0439	FE 74	t	CPI	74
043B	C2 2C 01	,	JNZ	012C
043E	3E 76	>v	MVI	A,76
0440	32 FD 09	2	STA	09FD
0443	3E 75	>u	MVI	A,75

H03B4

H03BD

0 command

0448	32 4E 0A	2N	STA	0A4E
044B	3E 12	>	MVI	A,12
044D	32 4D 0A	2M	STA	0A4D
0450	C3 6D 04	m	JMP	046D
0453	3A FD 09	!	LDA	09FD
0456	FE 73	s	CPI	73
0458	C2 2C 01	,	JNZ	012C
045B	3E 71	>a	MVI	A,71
045D	32 FD 09	2	STA	09FD
0460	3E 72	>r	MVI	A,72
0462	32 FF 09	2	STA	09FF
0465	32 4E 0A	2N	STA	0A4E
0468	3E 12	>	MVI	A,12
046A	32 4D 0A	2M	STA	0A4D
046D	3A 0D 0D	!	LDA	0D0D
0470	FE 4D	M	CPI	4D
0472	CA 72 00	r	JZ	0072
0475	FE 0D		CPI	0D
0477	CA D4 00		JZ	00D4
047A	C3 2C 01	,	JMP	012C
047D	2A 75 0A	*u	LHLD	0A75
0480	7D	}	MOV	A,L
0481	FE 36	6	CPI	36
0483	D2 6B 05	k	JNC	056B
0486	CD CF 04		CALL	04CF
0489	F5		PUSH	PSW
048A	2A 75 0A	*u	LHLD	0A75
048D	23	#	INX	H
048E	22 75 0A	"u	SHLD	0A75
0491	F1		POP	PSW
0492	D2 6B 05	k	JNC	056B
0495	CD 4F 05	0	CALL	054F
0498	CD B7 04		CALL	04B7
049B	CD 6D 08	m	CALL	086D
049E	3A 4D 0A	!M	LDA	0A4D
04A1	FE 00		CPI	00
04A3	C0		RNZ	
04A4	3E 02	>	MVI	A,02
04A6	32 4D 0A	2M	STA	0A4D
04A9	3A 4E 0A	!N	LDA	0A4E
04AC	EE 03		XRI	03
04AE	32 4E 0A	2N	STA	0A4E
04B1	32 78 0A	2x	STA	0A78
04B4	C3 98 04		JMP	0498
04B7	21 ED 09	!	LXI	H,09ED
04BA	3A 4D 0A	!M	LDA	0A4D
04BD	32 4F 0A	20	STA	0A4F
04C0	4F	0	MOV	C,A
04C1	06 00		MVI	B,00
04C3	09		DAD	B
04C4	7E	~	MOV	A,M
04C5	32 50 0A	2P	STA	0A50
04C8	3A 4E 0A	!N	LDA	0A4E
04CB	32 51 0A	2Q	STA	0A51
04CE	C9		RET	
04CF	CD 04 05		CALL	0504
04D2	CA EB 04		JZ	04EB
04D5	CD 1C 05		CALL	051C
04D8	21 77 0A	!w	LXI	H,0A77
04DB	35	5	DCR	M
04DC	F2 E7 04		JP	04E7
04DF	AF		XRA	A
04E0	26 00	&	MVI	H,00
04E2	2E F0	.	MVI	L,F0
04E4	22 75 0A	"u	SHLD	0A75

m'

CR

Label w/RET

04E7	DA CF 04		04CF
04EA	C9		RET
04EB	CD 29 05	)	CALL 0529
04EE	2A 75 0A	*u	LHLD 0A75
04F1	EB		XCHG
04F2	2A 79 0A	*w	LHLD 0A79
04F5	19		DAD D
04F6	7E	~	MOV A,M
04F7	32 4D 0A	2M	STA 0A4D
04FA	2A 7B 0A	*c	LHLD 0A7B
04FD	19		DAD D
04FE	7E	~	MOV A,M
04FF	32 4E 0A	2N	STA 0A4E
0502	37	7	STC
0503	C9		RET
0504	2A 75 0A	*u	LHLD 0A75
0507	EB		XCHG
0508	2A 7D 0A	*j	LHLD 0A7D
050B	19		DAD D
050C	3A 4D 0A	!M	LDA 0A4D
050F	BE		CMP M
0510	C2 1B 05		JNZ 051B
0513	2A 7F 0A	*	LHLD 0A7F
0516	19		DAD D
0517	3A 4E 0A	!N	LDA 0A4E
051A	BE		CMP M
051B	C9		RET
051C	2A 75 0A	*u	LHLD 0A75
051F	7D	j	MOV A,L
0520	C6 09		ADI 09
0522	6F	o	MOV L,A
0523	22 75 0A	"u	SHLD 0A75
0526	FE 36	6	CPI 36
0528	C9		RET
0529	2A 75 0A	*u	LHLD 0A75
052C	E5		PUSH H
052D	3A 77 0A	!w	LDA 0A77
0530	4F	0	MOV C,A
0531	3E FF	>	MVI A,FF
0533	32 77 0A	2w	STA 0A77
0536	CD 1C 05		CALL 051C
0539	21 77 0A	!w	LXI H,0A77
053C	34	4	INR M
053D	D2 4A 05	J	JNC 054A
0540	0D		DCR C
0541	FA 4A 05	J	JM 054A
0544	CD 04 05		CALL 0504
0547	CA 36 05	6	JZ 0536
054A	E1		POP H
054B	22 75 0A	"u	SHLD 0A75
054E	C9		RET
054F	3A 1E 07	!	LDA 071E
0552	3C	<	INR A
0553	57	w	MOV D,A
0554	06 05		MVI B,05
0556	0E FF		MVI C,FF
0558	3E FF	>	MVI A,FF
055A	3D	=	DCR A
055B	C2 5A 05	Z	JNZ 055A
055E	0D		DCR C
055F	C2 58 05	X	JNZ 0558
0562	05		DCR B
0563	C2 56 05	V	JNZ 0556
0566	15		DCR D
0567	C2 54 05	T	JNZ 0554
056A	C9		RET

056B	3E 0C			MVI	A,0C
056D	32 54 0A	2T		STA	0A54
0570	32 50 0A	2P		STA	0A50
0573	0E 14			MVI	C,14
0575	CD DF 05			CALL	05DF
0578	3E 04	>		MVI	A,04
057A	32 54 0A	2T		STA	0A54
057D	CD DD 05			CALL	05DD
0580	3A 50 0A	1P		LDA	0A50
0583	FE 0F			CPI	0F
0585	DA A3 05			JC	05A3
0588	21 ED 09	!		LXI	H,09ED
058B	06 00			MVI	B,00
058D	3A 4F 0A	1D		LDA	0A4F
0590	32 4D 0A	2M		STA	0A4D
0593	4F	0		MOV	C,A
0594	09			DAD	B
0595	7E	~		MOV	A,M
0596	32 50 0A	2P		STA	0A50
0599	3A 51 0A	1Q		LDA	0A51
059C	32 4E 0A	2N		STA	0A4E
059F	CD 6D 08	m		CALL	086D
05A2	C9			RET	
05A3	3E FF	>		MVI	A,FF
05A5	32 4F 0A	2D		STA	0A4F
05A8	32 50 0A	2P		STA	0A50
05AB	32 51 0A	2Q		STA	0A51
05AE	C9			RET	
05AF	21 0D 0A	!	H05AF	LXI	H,0A0D ✓
05B2	11 ED 09			LXI	D,09ED ✓
05B5	0E 20			MVI	C,20 ✓
05B7	7E	~	H05B7	MOV	A,M ✓
05B8	12			STAX	D ✓
05B9	23	#		INX	H ✓
05BA	13			INX	D ✓
05BB	0D			DCR	C ✓
05BC	C2 B7 05			JNZ	05B7 ✓
05BF	C9			RET	✓
05C0	CD AF 05		H05C0	CALL	05AF ✓
05C3	21 ED 09	!	H05C3	LXI	H,09ED ✓
05C6	11 FD 09			LXI	D,09FD ✓
05C9	0E 10			MVI	C,10 ✓
05CB	3E 77	>w	H05CB	MVI	A,77 ✓
05CD	96			SUB	M ✓
05CE	47	0		MOV	B,A ✓
05CF	EB			XCHG	
05D0	3E 77	>w		MVI	A,77 ✓
05D2	96			SUB	M ✓
05D3	70	P		MOV	M,B ✓
05D4	EB			XCHG	
05D5	77	w		MOV	M,A ✓
05D6	23	#		INX	H ✓
05D7	13			INX	D ✓
05D8	0D			DCR	C ✓
05D9	C2 CB 05			JNZ	05CB ✓
05DC	C9			RET	✓
05DD	0E 10			MVI	C,10
05DF	21 5D 0A	1J		LXI	H,0A5D
05E2	AF			XRA	A
05E3	77	w		MOV	M,A
05E4	23	#		INX	H
05E5	0D			DCR	C
05E6	F2 E3 05			JP	05E3
05E9	3E 10	>		MVI	A,10
05EB	32 4D 0A	2M		STA	0A4D
05EE	21 4D 0A	1M		LXI	H,0A4D

05F1	35		DLR	1
05F2	F8		RM	
05F3	CD 58 07	X	CALL	0758
05F6	3E 08	>	MVI	A,08
05F8	32 55 0A	2U	STA	0A55
05FB	3A 4D 0A	!M	LDA	0A4D
05FE	FE 08		CPI	08
0600	F2 56 06	V	JP	0656
0603	FE 06		CPI	06
0605	F2 43 06	C	JP	0643
0608	FE 04		CPI	04
060A	F2 35 06	5	JP	0635
060D	FE 01		CPI	01
060F	CA 1E 06		JZ	061E
0612	F2 27 06	,	JP	0627
0615	CD 98 06		CALL	0698
0618	C2 15 06		JNZ	0615
061B	C3 EE 05		JMP	05EE
061E	CD A9 06		CALL	06A9
0621	C2 1E 06		JNZ	061E
0624	C3 EE 05		JMP	05EE
0627	3E 04	>	MVI	A,04
0629	32 55 0A	2U	STA	0A55
062C	CD A9 06		CALL	06A9
062F	C2 2C 06	,	JNZ	062C
0632	C3 EE 05		JMP	05EE
0635	CD A9 06		CALL	06A9
0638	3A 55 0A	!U	LDA	0A55
063B	FE 04		CPI	04
063D	C2 35 06	5	JNZ	0635
0640	C3 EE 05		JMP	05EE
0643	3E 10	>	MVI	A,10
0645	32 55 0A	2U	STA	0A55
0648	CD 98 06		CALL	0698
064B	3A 55 0A	!U	LDA	0A55
064E	FE 08		CPI	08
0650	C2 48 06	H	JNZ	0648
0653	C3 EE 05		JMP	05EE
0656	3E 06	>	MVI	A,06
0658	32 55 0A	2U	STA	0A55
065B	CD DA 06		CALL	06DA
065E	FA 6B 06	k	JM	066B
0661	3A 52 0A	!R	LDA	0A52
0664	B7		ORA	A
0665	CA 6B 06	k	JZ	066B
0668	CD 93 07		CALL	0793
066B	CD 58 07	X	CALL	0758
066E	21 55 0A	!U	LXI	H,0A55
0671	35	5	DCR	M
0672	7E	~	MOV	A,M
0673	FE 05		CPI	05
0675	CA 5B 06	L	JZ	065B
0678	CD DA 06		CALL	06DA
067B	DA 8B 06		JC	068B
067E	FA EE 05		JM	05EE
0681	3A 52 0A	!R	LDA	0A52
0684	B7		ORA	A
0685	C2 EE 05		JNZ	05EE
0688	CD 93 07		CALL	0793
068B	3A 4E 0A	!N	LDA	0A4E
068E	E6 F0		ANI	F0
0690	FE 20		CPI	20
0692	CA 78 06	x	JZ	0678
0695	C3 EE 05		JMP	05EE
0698	CD DA 06		CALL	06DA
069B	FA A1 06		JM	06A1

069E	CD 93 07		CALL	0793
06A1	CD 58 07	X	CALL	0758
06A4	21 55 0A	1U	LXI	H,0A55
06A7	35	5	DCR	M
06A8	C9		RET	
06A9	CD DA 06		CALL	06DA
06AC	D2 C0 06		JNC	06C0
06AF	F5		PUSH	PSW
06B0	E1		POP	H
06B1	22 72 0A	*r	SHLD	0A72
06B4	3A 52 0A	1R	LDA	0A52
06B7	B7		ORA	A
06B8	CA A9 06		JZ	06A9
06BB	2A 72 0A	*r	LHLD	0A72
06BE	E5		PUSH	H
06BF	F1		POP	PSW
06C0	FA D2 06		JM	06D2
06C3	3A 52 0A	1R	LDA	0A52
06C6	F5		PUSH	PSW
06C7	CD 93 07		CALL	0793
06CA	F1		POP	PSW
06CB	32 52 0A	2R	STA	0A52
06CE	B7		ORA	A
06CF	CA A9 06		JZ	06A9
06D2	CD 58 07	X	CALL	0758
06D5	21 55 0A	1U	LXI	H,0A55
06D8	35	5	DCR	M
06D9	C9		RET	
06DA	3A 55 0A	1U	LDA	0A55
06DD	06 00		MVI	B,00
06DF	21 2C 0A	1r	LXI	H,0A2C
06E2	4F	0	MOV	C,A
06E3	09		DAD	B
06E4	3A 4E 0A	1N	LDA	0A4E
06E7	86		ADD	M
06E8	32 4E 0A	2N	STA	0A4E
06EB	E6 88		ANI	88
06ED	C2 51 07	Q	JNZ	0751
06F0	3A 4E 0A	1N	LDA	0A4E
06F3	0E 1F		MVI	C,1F
06F5	21 0C 0A	1	LXI	H,0A0C
06F8	BE		CMP	M
06F9	CA 04 07		JZ	0704
06FC	2B	+	DCX	H
06FD	0D		DCR	C
06FE	F2 F8 06		JP	06F8
0701	C3 12 07		JMP	0712
0704	79	u	MOV	A,C
0705	FE 10		CPI	10
0707	DA 51 07	Q	JC	0751
070A	3E 01	>	MVI	A,01
070C	32 52 0A	2R	STA	0A52
070F	C3 16 07		JMP	0716
0712	AF		XRA	A
0713	32 52 0A	2R	STA	0A52
0716	3A 54 0A	1T	LDA	0A54
0719	B7		ORA	A
071A	FA 4F 07	0	JM	074F
071D	FE 00		CPI	00
071F	F2 4F 07	0	JP	074F
0722	F5		PUSH	PSW
0723	3A 52 0A	1R	LDA	0A52
0726	F5		PUSH	PSW
0727	3E F9	>	MVI	A,F9
0729	32 54 0A	2T	STA	0A54
072C	32 53 0A	2S	STA	0A53

072F	CD 6D 08	m	CALL	086D
0732	CD C3 05		CALL	05C3
0735	CD E9 05		CALL	05E9
0738	CD 70 07	P	CALL	0770
073B	F1		POP	PSW
073C	32 52 0A	2R	STA	0A52
073F	F1		POP	PSW
0740	32 54 0A	2T	STA	0A54
0743	3A 53 0A	IS	LDA	0A53
0746	B7		ORA	A
0747	FA 4F 07	0	JM	074F
074A	3E 80	>	MVI	A,80
074C	B7		ORA	A
074D	37	7	STC	
074E	C9		RET	
074F	AF		XRA	A
0750	C9		RET	
0751	AF		XRA	A
0752	32 52 0A	2R	STA	0A52
0755	2F	/	CMA	
0756	B7		ORA	A
0757	C9		RET	
0758	21 ED 09	I	LXI	H,09ED
075B	06 00		MVI	B,00
075D	3A 4D 0A	IM	LDA	0A4D
0760	4F	0	MOV	C,A
0761	09		DAD	B
0762	7E	~	MOV	A,M
0763	32 4E 0A	2N	STA	0A4E
0764	C9		RET	
0767	CD 6D 08	m	CALL	086D
076A	CD C3 05		CALL	05C3
076D	CD E9 05		CALL	05E9
0770	CD C3 05		CALL	05C3
0773	21 00 00	I	LXI	H,0000
0776	39	9	DAD	SP
0777	22 57 0A	*W	SHLD	0A57
077A	2A 59 0A	*Y	LHLD	0A59
077D	F9		SPHL	
077E	D1		POP	D
077F	21 55 0A	IU	LXI	H,0A55
0782	72	r	MOV	M,D
0783	21 4D 0A	IM	LXI	H,0A4D
0786	73	s	MOV	M,E
0787	C1		POP	B
0788	E1		POP	H
0789	71	g	MOV	M,C
078A	E1		POP	H
078B	70	P	MOV	M,B
078C	78	x	MOV	A,B
078D	32 4E 0A	2N	STA	0A4E
0790	C3 60 08		JMP	08A0
0793	3A 54 0A	IT	LDA	0A54
0796	B7		ORA	A
0797	16 00		MVI	D,00
0799	5F		MOV	E,A
079A	FA 20 08		JM	0820
079D	3A 4D 0A	IM	LDA	0A4D
07A0	B7		ORA	A
07A1	CA B2 07		JZ	07B2
07A4	47	G	MOV	B,A
07A5	3E 08	>	MVI	A,08
07A7	BB		CMP	E
07A8	C2 B2 07		JNZ	07B2
07AB	3A 65 0A	ie	LDA	0A65
07AE	BB		CMP	B

07AF	CA FB 07		JZ	07FB
07B2	21 62 0A	1b	LXI	H,0A62
07B5	19		DAD	D
07B6	34	4	INR	M
07B7	3E 01	>	MVI	A,01
07B9	B8		CMP	B
07BA	C2 BE 07		JNZ	07BE
07BD	34	4	INR	M
07BE	3A 52 0A	!R	LDA	0A52
07C1	B7		ORA	A
07C2	CA F2 07		JZ	07F2
07C5	0E 0F		MVI	C,0F
07C7	21 0C 0A	!	LXI	H,0A0C
07CA	3A 4E 0A	!N	LDA	0A4E
07CD	BE		CMP	M
07CE	CA D7 07		JZ	07D7
07D1	2B	+	DCX	H
07D2	0D		DCR	C
07D3	F2 CD 07		JP	07CD
07D6	76	v	HLT	
07D7	21 3D 0A	!=	LXI	H,0A3D
07DA	06 00		MVI	B,00
07DC	09		DAD	B
07DD	7E	~	MOV	A,M
07DE	21 63 0A	!c	LXI	H,0A63
07E1	19		DAD	D
07E2	BE		CMP	M
07E3	DA EC 07		JC	07EC
07E6	77	w	MOV	M,A
07E7	21 65 0A	!e	LXI	H,0A65
07EA	19		DAD	D
07EB	71	a	MOV	M,C
07EC	21 64 0A	!d	LXI	H,0A64
07EF	19		DAD	D
07F0	86		ADD	M
07F1	77	w	MOV	M,A
07F2	7B	c	MOV	A,E
07F3	FE 04		CPI	04
07F5	CA FC 07		JZ	07FC
07F8	FA 34 08	4	JM	0834
07FB	C9		RET	
07FC	3A 67 0A	!d	LDA	0A67
07FF	32 5C 0A	2\	STA	0A5C
0802	AF		XRA	A
0803	32 54 0A	2T	STA	0A54
0806	CD 6D 08	m	CALL	086D
0809	CD C3 05		CALL	05C3
080C	CD DD 05		CALL	05DD
080F	CD C3 05		CALL	05C3
0812	3E 08	>	MVI	A,08
0814	32 54 0A	2T	STA	0A54
0817	CD E9 05		CALL	05E9
081A	CD 73 07	s	CALL	0773
081D	C3 AC 08		JMP	08AC
0820	16 FF		MVI	D,FF
0822	FE F9		CPI	F9
0824	C2 34 08	4	JNZ	0834
0827	3A FD 09	!	LDA	09FD
082A	21 4E 0A	!N	LXI	H,0A4E
082D	BE		CMP	M
082E	C0		RNZ	
082F	AF		XRA	A
0830	32 53 0A	2S	STA	0A53
0833	C9		RET	
0834	3A 52 0A	!R	LDA	0A52
0837	B7		ORA	A

← HLT

0838	C8				RZ	
0839	3A 4E 0A	:N			LDA	0A4E
083C	0E 07				MVI	C,07
083E	21 04 0A	I			LXI	H,0A04
0841	BE				CMP	M
0842	CA 4B 08	K			JZ	084B
0845	2B	+			DCX	H
0846	0D				DCR	C
0847	C2 41 08	A			JNZ	0841
084A	C9				RET	
084B	21 3D 0A	I=			LXI	H,0A3D
084E	06 00				MVI	B,00
0850	09				DAD	B
0851	7E	~			MOV	A,M
0852	21 61 0A	Ia			LXI	H,0A61
0855	19				DAD	D
0856	BE				CMP	M
0857	DA 5B 08	C			JC	085B
085A	77	w			MOV	M,A
085B	1B				DCX	D
085C	7B	C			MOV	A,E
085D	32 54 0A	2T			STA	0A54
0860	FE FB				CPI	FB
0862	CA 68 08	h			JZ	0868
0865	CD 67 07	g			CALL	0767
0868	21 54 0A	IT			LXI	H,0A54
086B	34	4			INR	M
086C	C9				RET	
086D	21 00 00	I	H086D		LXI	H,0000
0870	39	9			DAD	SP
0871	22 57 0A	*W			SHLD	0A57
0874	2A 59 0A	*Y			LHLD	0A59
0877	F9				SPHL	
0878	3A 4E 0A	:N			LDA	0A4E
087B	47	G			MOV	B,A
087C	0E 1F				MVI	C,1F
087E	21 0C 0A	I			LXI	H,0A0C
0881	BE		H0881		CMP	M
0882	CA 8A 08				JZ	088A
0885	2B	+			DCX	H
0886	0D				DCR	C
0887	F2 81 08				JP	0881
088A	36 CC	6	H088A		MVI	M,CC
088C	E5				PUSH	H
088D	21 ED 09	I			LXI	H,09ED
0890	16 00				MVI	D,00
0892	3A 4D 0A	:M			LDA	0A4D
0895	5F				MOV	E,A
0896	19				DAD	D
0897	E5				PUSH	H
0898	4E	N			MOV	C,M
0899	70	P			MOV	M,B
089A	C5				PUSH	B
089B	21 55 0A	IU			LXI	H,0A55
089E	56	V			MOV	D,M
089F	D5				PUSH	D
08A0	21 00 00	I	STAKRELIEF		LXI	H,0000
08A3	39	9			DAD	SP
08A4	22 59 0A	*Y			SHLD	0A59
08A7	2A 57 0A	*W			LHLD	0A57
08AA	F9				SPHL	
08AB	C9				RET	
08AC	97				SUB	A
08AD	3E 80	>			MVI	A,80
08AF	21 6A 0A	IJ			LXI	H,0A6A
08B2	86				ADD	M

08B3	86	*	INX	H
08B4	86		ADD	M
08B5	23	#	INX	H
08B6	86		ADD	M
08B7	21 60 0A	I'	LXI	H, 0A60
08BA	86		ADD	M
08BB	21 5E 0A	I''	LXI	H, 0A5E
08BE	86		ADD	M
08BF	21 6F 0A	I ₀	LXI	H, 0A6F
08C2	96		SUB	M
08C3	23	#	INX	H
08C4	96		SUB	M
08C5	21 61 0A	I _a	LXI	H, 0A61
08C8	96		SUB	M
08C9	21 5F 0A	I	LXI	H, 0A5F
08CC	96		SUB	M
08CD	21 5D 0A	I _j	LXI	H, 0A5D
08D0	96		SUB	M
08D1	21 6E 0A	I _n	LXI	H, 0A6E
08D4	96		SUB	M
08D5	21 62 0A	I _b	LXI	H, 0A62
08D8	96		SUB	M
08D9	D2 DD 08		JNC	08DD
08DC	97		SUB	A
08DD	1F		RAR	
08DE	C6 40	@	ADI	40
08E0	21 6B 0A	I _k	LXI	H, 0A6B
08E3	86		ADD	M
08E4	23	#	INX	H
08E5	86		ADD	M
08E6	21 63 0A	I _c	LXI	H, 0A63
08E9	96		SUB	M
08EA	1F		RAR	
08EB	C6 90		ADI	90
08ED	21 5C 0A	I\	LXI	H, 0A5C
08F0	86		ADD	M
08F1	86		ADD	M
08F2	86		ADD	M
08F3	86		ADD	M
08F4	21 60 0A	I'	LXI	H, 0A60
08F7	86		ADD	M
08F8	21 63 0A	I _c	LXI	H, 0A63
08FB	96		SUB	M
08FC	96		SUB	M
08FD	23	#	INX	H
08FE	96		SUB	M
08FF	96		SUB	M
0900	21 5F 0A	I	LXI	H, 0A5F
0903	96		SUB	M
0904	F5		PUSH	PSW
0905	3A 4E 0A	I _N	LDA	0A4E
0908	FE 33	3	CPI	33
090A	CA 30 09	0	JZ	0930
090D	FE 34	4	CPI	34
090F	CA 30 09	0	JZ	0930
0912	FE 22	"	CPI	22
0914	CA 30 09	0	JZ	0930
0917	FE 25	%	CPI	25
0919	CA 30 09	0	JZ	0930
091C	3A 4D 0A	I _M	LDA	0A4D
091F	B7		ORA	A
0920	CA 34 09	4	JZ	0934
0923	21 ED 09	I	LXI	H, 09ED
0926	06 00		MVI	B, 00
0928	4F	0	MOV	C, A
0929	09		DAD	B

7 bytes 2 bytes
 { LXI H, 0A60
 ADD M
 LXI H, 0A5E
 ADD M
 LXI H, 0A6F
 ADD M
 LXI H, 0A61
 ADD M
 LXI H, 0A5F
 ADD M
 LXI H, 0A5D
 ADD M
 LXI H, 0A6E
 ADD M
 LXI H, 0A62
 ADD M
 JNC 08DD
 SUB A
 RAR
 ADI 40
 LXI H, 0A6B
 ADD M
 INX H
 ADD M
 LXI H, 0A63
 SUB M
 RAR
 ADI 90
 LXI H, 0A5C
 ADD M
 ADD M
 ADD M
 ADD M
 LXI H, 0A60
 ADD M
 LXI H, 0A63
 SUB M
 SUB M
 INX H
 SUB M
 SUB M
 LXI H, 0A5F
 SUB M
 PUSH PSW
 LDA 0A4E
 CPI 33
 JZ 0930
 CPI 34
 JZ 0930
 CPI 22
 JZ 0930
 CPI 25
 JZ 0930
 LDA 0A4D
 ORA A
 JZ 0934
 LXI H, 09ED
 MVI B, 00
 MOV C, A
 DAD B

092A	7E		
092B	FE 10		
092D	F2 34 09	4	
0930	F1		
0931	C6 02		
0933	F5		
0934	3A 63 0A	:c	
0937	21 3D 0A	l=	
093A	BE		
093B	C2 43 09	C	
093E	F1		
093F	97		
0940	C3 5B 09	C	
0943	3A 62 0A	:b	
0946	B7		
0947	C2 5A 09	Z	
094A	3A 6D 0A	:m	
094D	B7		
094E	C2 5A 09	Z	
0951	F1		
0952	3E FF	>	
0954	32 81 0A	2	
0957	C3 5B 09	C	
095A	F1		
095B	0E 04		
095D	21 54 0A	IT	
0960	71	a	
0961	21 50 0A	IP	
0964	32 5B 0A	2C	
0967	BE		
0968	DA 8E 09		
096B	CA 8E 09		
096E	32 50 0A	2F	
0971	3A 4D 0A	:M	
0974	32 4F 0A	20	
0977	3A 4E 0A	:N	
097A	32 51 0A	2Q	
097D	AF		
097E	32 74 0A	2t	
0981	3A 6B 0A	:k	
0984	21 3D 0A	l=	
0987	BE		
0988	C2 8E 09		
098B	32 74 0A	2t	
098E	C9		

MOV	A,M
CPI	10
JP	0934
POP	PSW
ADI	02
PUSH	PSW
LDA	0A63
LXI	H,0A3D
CMP	M
JNZ	0943
POP	PSW
SUB	A
JMP	095B
LDA	0A62
ORA	A
JNZ	095A
LDA	0A6D
ORA	A
JNZ	095A
POP	PSW
MVI	A,FF
STA	0A81
JMP	095B
POP	PSW
MVI	C,04
LXI	H,0A54
MOV	M,C
LXI	H,0A50
STA	0A5B
CMP	M
JC	098E
JZ	098E
STA	0A50
LDA	0A4D
STA	0A4F
LDA	0A4E
STA	0A51
XRA	A
STA	0A74
LDA	0A6B
LXI	H,0A3D
CMP	M
JNZ	098E
STA	0A74
RET	

098F	21 DD 09	!	
0992	47	G	
0993	1F		
0994	1F		
0995	1F		
0996	1F		
0997	CD A2 09		
099A	77	w	
099B	23	#	
099C	78	x	
099D	CD A2 09		
09A0	77	w	
09A1	C9		
09A2	E6 0F		
09A4	C6 30	0	
09A6	FE 3A	:	
09A8	DB		
09A9	C6 07		
09AB	C9		

LXI	H,09DD
MOV	B,A
RAR	
RAR	
RAR	
RAR	
CALL	09A2
MOV	M,A
INX	H
MOV	A,B
CALL	09A2
MOV	M,A
RET	
ANI	0F
ADI	30
CPI	3A
RC	
ADI	07
RET	

09AC	06 0D		
09AE	CD BF 09		

MVI	B,0D
CALL	09BF

CRLF R10

09B1 08 0A
09B3 CD BF 09
09B6 06 7F
09B8 CD BF 09
09BB CD BF 09
09BE C9

MVI 090A
CALL 09BF
MVI B, 7F
CALL 09BF
CALL 09BF
RET

CROUT

09BF E5
09C0 C5
09C1 D5
09C2 CD D7 09
09C5 D1
09C6 C1
09C7 E1
09C8 C9

COUT

WHI

PUSH H
PUSH B
PUSH D
CALL 09D7
POP D
POP B
POP H
RET

OUTPUT a char

09C9 E5
09CA D5
09CB C5
09CC CD DA 09
09CF 78
09D0 C1
09D1 D1
09D2 E1
09D3 E6 7F
09D5 47
09D6 C9

COUT

x WHI

PUSH H
PUSH D
PUSH B
CALL 09DA
MOV A, B
POP B
POP D
POP H
ANI 7F
MOV B, A
RET

INPUT a char ?

09D7 C3 77 C0
09DA C3 F1 0D

w

JMP C077
JMP 0DF1

INPUT WHI WHI
OUTPUT WHI WHI

09DD 33
09DE 34
09DF 00
09E0 00
09E1 00
09E2 00
09E3 00
09E4 00
09E5 00
09E6 00
09E7 00
09E8 00
09E9 00
09EA 00
09EB 00

13

INX SP
INR M
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP

09EC CC 04 03

H09ED

CZ 0304

09EF 07
09F0 00
09F1 05
09F2 02
09F3 06 01
09F5 17
09F6 10
09F7 16 11
09F9 15
09FA 12
09FB 13
09FC 14
09FD 74
09FE 73
09FF 77
0A00 70
0A01 75
0A02 72
0A03 76
0A04 71
0A05 67
0A06 60
0A07 66

t
e
w
p
u
r
v
g
e
'
f

RLC
NOP
DCR B
STAX B
MVI B, 01
RAL
DB
MVI D, 11
DCR D
STAX D
INX D
INR D
MOV M, H
MOV M, E
MOV M, A
MOV M, B
MOV M, L
MOV M, D
HLT
MOV M, C
MOV H, A
MOV H, B
MOV H, M

0A07 66

0A08	61	e	MOV	H,C
0A09	65	e	MOV	H,L
0A0A	62	b	MOV	H,D
0A0B	63	c	MOV	H,E
0A0C	64	d	MOV	H,H
0A0D	03	H0A0D		INX B
0A0E	04		INR	B
0A0F	00		NOP	
0A10	07		RLC	
0A11	02		STAX	B
0A12	05		DCR	B
0A13	01 06 10		LXI	B,1006
0A16	17		RAL	
0A17	11 16 12		LXI	D,1216
0A1A	15		DCR	D
0A1B	14		INR	D
0A1C	13		INX	D
0A1D	73	s	MOV	M,E
0A1E	74	t	MOV	M,H
0A1F	70	p	MOV	M,B
0A20	77	w	MOV	M,A
0A21	72	r	MOV	M,D
0A22	75	u	MOV	M,L
0A23	71	g	MOV	M,C
0A24	76	v	HLT	
0A25	60	,	MOV	H,B
0A26	67	e	MOV	H,A
0A27	61	a	MOV	H,C
0A28	66	f	MOV	H,M
0A29	62	b	MOV	H,D
0A2A	65	e	MOV	H,L
0A2B	64	d	MOV	H,H
0A2C	63	c	MOV	H,E
0A2D	F0		RP	
0A2E	FF		RST	7
0A2F	01 10 11		LXI	B,1110
0A32	0F		RRC	
0A33	EF		RST	5
0A34	F1		POP	PSW
0A35	DF		RST	3
0A36	E1		POP	H
0A37	EE F2		XRI	F2
0A39	12		STAX	D
0A3A	0E 1F		MVI	C,1F
0A3C	21 0B 0A	I	LXI	H,0A0B
0A3F	06 06		MVI	B,06
0A41	04		INR	B
0A42	04		INR	B
0A43	04		INR	B
0A44	04		INR	B
0A45	02		STAX	B
0A46	02	8	STAX	B
0A47	02		STAX	B
0A48	02		STAX	B
0A49	02		STAX	B
0A4A	02		STAX	B
0A4B	02		STAX	B
0A4C	02		STAX	B
0A4D	EE EE		XRI	EE
0A4F	0F		RRC	
0A50	14		INR	D
0A51	34	4	INR	M
0A52	00		NOP	
0A53	00		NOP	
0A54	04		INR	B
0A55	00		NOP	

0AA6	1D		DCR	E
0AA7	1E 1D		MVI	E,1D
0AA9	16 1A		MVI	D,1A
0AAB	14		INR	D
0AAC	12		STAX	D
0AAD	17		RAL	
0AAE	11 11 1F		LXI	D,1F11
0AB1	16 17		MVI	D,17
0AB3	14		INR	D
0AB4	12		STAX	D
0AB5	1E 15		MVI	E,15
0AB7	1B		DCX	D
0AB8	12		STAX	D
0AB9	1F		RAR	
0ABA	16 14		MVI	D,14
0ABC	12		STAX	D
0ABD	1E 11		MVI	E,11
0ABF	14		INR	D
0AC0	1E 17		MVI	E,17
0AC2	1F		RAR	
0AC3	16 14		MVI	D,14
0AC5	1D		DCR	E
0AC6	1E 1D		MVI	E,1D
0AC8	17		RAL	
0AC9	12		STAX	D
0ACA	1B		DCX	D
0ACB	EE 43	C	XRI	43
0ACD	55	U	MOV	D,L
0ACE	52	R	MOV	D,D
0ACF	33	3	INX	SP
0AD0	63	c	MOV	H,E
0AD1	54	T	MOV	D,H
0AD2	55	U	MOV	D,L
0AD3	66	f	MOV	H,M
0AD4	EE 53	S	XRI	53
0AD6	44	D	MOV	B,H
0AD7	52	R	MOV	D,D
0AD8	63	c	MOV	H,E
0AD9	64	d	MOV	H,H
0ADA	63	c	MOV	H,E
0ADB	72	r	MOV	M,D
0ADC	45	E	MOV	B,L
0ADD	43	C	MOV	B,E
0ADE	42	B	MOV	B,D
0ADF	55	U	MOV	D,L
0AE0	56	V	MOV	D,M
0AE1	66	f	MOV	H,M
0AE2	75	u	MOV	M,L
0AE3	52	R	MOV	D,D
0AE4	62	b	MOV	H,D
0AE5	52	R	MOV	D,D
0AE6	44	D	MOV	B,H
0AE7	55	U	MOV	D,L
0AE8	52	R	MOV	D,D
0AE9	31 75 53	1uS	LXI	SP,5375
0AEC	36 52	6R	MVI	M,52
0AEE	74	t	MOV	M,H
0AEF	44	D	MOV	B,H
0AF0	55	U	MOV	D,L
0AF1	31 75 43	1uC	LXI	SP,4375
0AF4	64	d	MOV	H,H
0AF5	22 34 52	*4R	SHLD	5234
0AF8	44	D	MOV	B,H
0AF9	55	U	MOV	D,L
0AFA	42	B	MOV	B,D
0AFB	52	R	MOV	D,D

0AFD	43	C	MOV	B,E
0AFE	52	R	MOV	D,D
0AFF	75	U	MOV	M,L
0B00	52	R	MOV	D,D
0B01	0F		RRC	
0B02	06 04		MVI	B,04
0B04	00		NOP	
0B05	0E 01		MVI	C,01
0B07	04		INR	B
0B08	0E 07		MVI	C,07
0B0A	0F		RRC	
0B0B	0E 07		MVI	C,07
0B0D	05		DCR	B
0B0E	0F		RRC	
0B0F	05		DCR	B
0B10	01 0C 06		LXI	B,060C
0B13	06 0F		MVI	B,0F
0B15	0B		DCX	B
0B16	05		DCR	B
0B17	04		INR	B
0B18	00		NOP	
0B19	06 06		MVI	B,06
0B1B	0C		INR	C
0B1C	0F		RRC	
0B1D	07		RLC	
0B1E	06 04		MVI	B,04
0B20	00		NOP	
0B21	0E 04		MVI	C,04
0B23	01 07 0F		LXI	B,0F07
0B26	07		RLC	
0B27	06 06		MVI	B,06
0B29	04		INR	B
0B2A	06 0B		MVI	B,0B
0B2C	06 00		MVI	B,00
0B2E	0F		RRC	
0B2F	07		RLC	
0B30	04		INR	B
0B31	06 0F		MVI	B,0F
0B33	04		INR	B
0B34	06 06		MVI	B,06
0B36	04		INR	B
0B37	33	3	INX	SP
0B38	22 46 01	"F	SHLD	0146
0B3B	34	4	INR	M
0B3C	13		INX	D
0B3D	55	U	MOV	D,L
0B3E	43	C	MOV	B,E
0B3F	25	%	DCR	H
0B40	33	3	INX	SP
0B41	34	4	INR	M
0B42	25	%	DCR	H
0B43	41	A	MOV	B,C
0B44	43	C	MOV	B,E
0B45	63	c	MOV	H,E
0B46	14		INR	D
0B47	32 22 25	2*%	STA	2522
0B4A	24	#	INR	H
0B4B	21 11 14	I	LXI	H,1411
0B4E	06 44	D	MVI	B,44
0B50	52	R	MOV	D,D
0B51	35	5	DCR	M
0B52	34	4	INR	M
0B53	22 25 41	"%A	SHLD	4125
0B56	06 23	#	MVI	B,23
0B58	52	R	MOV	D,D

OB57	14			INR	B
OB5A	03			INX	M
OB5B	34		4	SHLD	4425
OB5C	22	25	44	INR	D
OB5F	14			INX	H
OB60	23		#	SHLD	0611
OB61	22	11	06	INR	M
OB64	34		4	SHLD	2532
OB65	22	32	25	MOV	B,E
OB68	43		C	MOV	B,C
OB69	41		A	MOV	B,H
OB6A	44		D	MOV	D,D
OB6B	52		R	MOV	D,D
OB6C	52		R	MOV	C,L
OB6D	4D		M	MOV	C,C
OB6E	49		I	MOV	B,E
OB6F	43		C	MOV	D,D
OB70	52		R	MOV	C,A
OB71	4F		O	MOV	B,E
OB72	43		C	MOV	C,B
OB73	48		H	MOV	B,L
OB74	45		E	MOV	D,E
OB75	53		S	MOV	D,E
OB76	53		S	DB	
OB77	20			DB	
OB78	20			DB	
OB79	20			DB	
OB7A	20			DB	
OB7B	20			DB	
OB7C	20			DB	
OB7D	20			DB	
OB7E	20			DB	
OB7F	20			DB	
OB80	20			DB	
OB81	20			DB	
OB82	20			DB	
OB83	20			DB	
OB84	20			DB	
OB85	20			DB	
OB86	20			DB	
OB87	20			DB	
OB88	20			DB	
OB89	28		(	MOV	B,E
OB8A	43		C	DAD	H
OB8B	29		)	DB	
OB8C	20			LXI	SP,3739
OB8D	31	39	37	STC	
OB90	37		7	MVI	L,0D
OB91	2E	0D	.	MOV	D,A
OB93	57		W	MOV	D,D
OB94	52		R	MOV	C,C
OB95	49		I	MOV	D,H
OB96	54		T	MOV	D,H
OB97	54		T	MOV	B,L
OB98	45		E	MOV	C,M
OB99	4E		N	DB	
OB9A	20			MOV	B,D
OB9B	42		B	MOV	E,C
OB9C	59		Y	LDA	5020
OB9D	3A	20	50	MVI	L,20
OBA0	2E	20	.	MOV	C,D
OBA2	4A		J	MOV	B,L
OBA3	45		E	MOV	C,M
OBA4	4E		N	MOV	C,M
OBA5	4E		N	MOV	C,C
OBA6	49		I		

DB 'MICROCHESS'

(C) 1977. Cr

WRITTEN BY

: P. JENNINGS

ST. O'BRIEN. CR

OBED	53	S	MOV	D,E
OBEE	2C		INR	L
OB EF	42	B	MOV	B,D
OBFO	2C		INR	L
OBFI	4E	N	MOV	C,M
OBF2	29	)	DAD	H
OBF3	20		DB	
OBF4	0D		DCR	C
OBF5	44	D	MOV	B,H
OBF6	4F	O	MOV	C,A
OBF7	20		DB	
OBF8	59	Y	MOV	E,C
OBF9	4F	O	MOV	C,A
OBFA	55	U	MOV	D,L
OBFB	20		DB	
OBFC	57	W	MOV	D,A
OBFD	41	A	MOV	B,C
OBFE	4E	N	MOV	C,M
OBFF	54	T	MOV	D,H
OC00	20		DB	
OC01	57	W	MOV	D,A
OC02	48	H	MOV	C,B
OC03	49	I	MOV	C,C
OC04	54	T	MOV	D,H
OC05	45	E	MOV	B,L
OC06	20		DB	
OC07	3F	?	CMC	
OC08	20		DB	
OC09	28	(	DB	
OC0A	59	Y	MOV	E,C
OC0B	2C		INR	L
OC0C	4E	N	MOV	C,M
OC0D	29	)	DAD	H
OC0E	20		DB	
OC0F	0D		DCR	C
OC10	3A 20 20	:	LDA	2020
OC13	20		DB	
OC14	20		DB	
OC15	0D		DCR	C
OC16	4D	M	MOV	C,L
OC17	43	C	MOV	B,E
OC18	20		DB	
OC19	3A 20 31	: 1	LDA	3120
OC1C	34	4	INR	M
OC1D	2D	-	DCR	L
OC1E	33	3	INX	SP
OC1F	34	4	INR	M
OC20	20		DB	
OC21	20		DB	
OC22	0D		DCR	C
OC23	49	I	MOV	C,C
OC24	4E	N	MOV	C,M
OC25	50	F	MOV	D,B
OC26	55	U	MOV	D,L
OC27	54	T	MOV	D,H
OC28	20		DB	
OC29	45	E	MOV	B,L
OC2A	52	R	MOV	D,D
OC2B	52	R	MOV	D,D
OC2C	4F	O	MOV	C,A
OC2D	52	R	MOV	D,D
OC2E	2E 0D	.	MVI	L,0D
OC30	43	C	MOV	B,E
OC31	48	H	MOV	C,B
OC32	45	E	MOV	B,L

Super Blitz

Blitz

Novice

HPC23

HPC30

P2

0C33	45	C	MOV	D,E
0C34	4B	K	MOV	C,E
0C35	4D	M	MOV	C,L
0C36	41	A	MOV	B,C
0C37	54	T	MOV	D,H
0C38	45	E	MOV	B,L
0C39	20		DB	
0C3A	2D	-	DCR	L
0C3B	20		DB	
0C3C	59	Y	MOV	E,C
0C3D	4F	O	MOV	C,A
0C3E	55	U	MOV	D,L
0C3F	20		DB	
0C40	57	W	MOV	D,A
0C41	49	I	MOV	C,C
0C42	4E	N	MOV	C,M
0C43	21 21 21	!!!	LXI	H,2121
0C44	0D		DCR	C

VP2

0C47	50	P	MOV	D,B
0C48	4C	L	MOV	C,H
0C49	41	A	MOV	B,C
0C4A	59	Y	MOV	E,C
0C4B	20		DB	
0C4C	41	A	MOV	B,C
0C4D	47	G	MOV	B,A
0C4E	41	A	MOV	B,C
0C4F	49	I	MOV	C,C
0C50	4E	N	MOV	C,M
0C51	20		DB	
0C52	3F	T	CMC	
0C53	20		DB	
0C54	28	(	DB	
0C55	59	Y	MOV	E,C
0C56	2C	,	INR	L
0C57	4E	N	MOV	C,M
0C58	29	)	DAD	H
0C59	20		DB	
0C5A	0D		DCR	C

VP2

0C5B	54	T	MOV	D,H
0C5C	48	H	MOV	C,B
0C5D	41	A	MOV	B,C
0C5E	4E	N	MOV	C,M
0C5F	4B	K	MOV	C,E
0C60	53	S	MOV	D,E
0C61	20		DB	
0C62	46	F	MOV	B,M
0C63	4F	O	MOV	C,A
0C64	52	R	MOV	D,D
0C65	20		DB	
0C66	54	T	MOV	D,H
0C67	48	H	MOV	C,B
0C68	45	E	MOV	B,L
0C69	20		DB	
0C6A	47	G	MOV	B,A
0C6B	41	A	MOV	B,C
0C6C	4D	M	MOV	C,L
0C6D	45	E	MOV	B,L
0C6E	2E 2E	..	MVI	L,2E
0C70	2E 20	.	MVI	L,20
0C72	4D	M	MOV	C,L
0C73	49	I	MOV	C,C
0C74	43	C	MOV	B,E
0C75	52	R	MOV	D,D
0C76	4F	O	MOV	C,A
0C77	43	C	MOV	B,E
0C78	48	H	MOV	C,B

VP2

OC7A	53	S	MOV	D,E	VP2
OC7B	53	S	MOV	D,E	
OC7C	2E 0D	.	MVI	L,0D	
OC7E	59	Y	MOV	E,C	
OC7F	4F	O	MOV	C,A	
OC80	55	U	MOV	D,L	
OC81	20		DB		
OC82	52	R	MOV	D,D	
OC83	45	E	MOV	B,L	
OC84	53	S	MOV	D,E	
OC85	49	I	MOV	C,C	
OC86	47	G	MOV	B,A	
OC87	4E	N	MOV	C,M	
OC88	45	E	MOV	B,L	
OC89	44	D	MOV	B,H	
OC8A	20		DB		VP2
OC8B	2D	-	DCR	L	
OC8C	20		DB		
OC8D	49	I	MOV	C,C	
OC8E	20		DB		
OC8F	57	W	MOV	D,A	
OC90	49	I	MOV	C,C	
OC91	4E	N	MOV	C,M	
OC92	21 21 21	!!!	LXI	H,2121	
OC95	0D		DCR	C	
OC96	20		DB		
OC97	20		DB		
OC98	43	C	MOV	B,E	
OC99	48	H	MOV	C,B	
OC9A	45	E	MOV	B,L	
OC9B	43	C	MOV	B,E	
OC9C	4B	K	MOV	C,E	
OC9D	4D	M	MOV	C,L	
OC9E	41	A	MOV	B,C	
OC9F	54	T	MOV	D,H	
OCA0	45	E	MOV	B,L	VP2
OCA1	20		DB		
OCA2	2D	-	DCR	L	
OCA3	20		DB		
OCA4	49	I	MOV	C,C	
OCA5	20		DB		
OCA6	57	W	MOV	D,A	
OCA7	49	I	MOV	C,C	
OCA8	4E	N	MOV	C,M	
OCA9	21 21 21	!!!	LXI	H,2121	
OCAE	21 0D 20	HOCAE	LXI	H,200D	
OCAF	43	C	MOV	B,E	
OCB0	48	H	MOV	C,B	
OCB1	45	E	MOV	B,L	
OCB2	43	C	MOV	B,E	
OCB3	4B	K	MOV	C,E	
OCB4	21 0D 2B	! +	LXI	H,2B0D	VP2
OCB7	2D	-	DCR	L	
OCB8	2D	-	DCR	L	
OCB9	2D	-	DCR	L	
OCBA	2D	-	DCR	L	
OCBB	2D	-	DCR	L	
OCBC	2D	-	DCR	L	
OCBD	20		DB		
OCBE	4D	M	MOV	C,L	
OCBF	49	I	MOV	C,C	
OCC0	43	C	MOV	B,E	
OCC1	52	R	MOV	D,D	
OCC2	4F	O	MOV	C,A	
OCC3	43	C	MOV	B,E	

OCC4	4B	H	MOV	B, B
OCC5	45	E	MOV	B, L
OCC6	53	S	MOV	D, E
OCC7	53	S	MOV	D, E
OCC8	20		DB	
OCC9	2D	-	DCR	L
OCCA	2D	-	DCR	L
OCCB	2D	-	DCR	L
OCCC	2D	-	DCR	L
OCCD	2D	-	DCR	L
OCCE	2D	-	DCR	L
OCCF	2D	-	DCR	L
OCD0	2B	+	DCX	H
OCD1	0D		DCR	C
OCD2	21 20 20	!	LXI	H, 2020
OCD5	20		DB	
OCD6	20		DB	
OCD7	20		DB	
OCD8	20		DB	
OCD9	20		DB	
OCDA	20		DB	
OCDB	20		DB	
OCDC	20		DB	
OCDD	20		DB	
OCDE	20		DB	
OCDF	20		DB	
OCE0	20		DB	
OCE1	20		DB	
OCE2	20		DB	
OCE3	20		DB	
OCE4	20		DB	
OCE5	20		DB	
OCE6	20		DB	
OCE7	20		DB	
OCE8	20		DB	
OCE9	20		DB	
OCEA	20		DB	
OCEB	20		DB	
OCEC	21 0D 2B	! +	LXI	H, 2B0D
OCEF	2D	-	DCR	L
OCF0	2D	-	DCR	L
OCF1	2D	-	DCR	L
OCF2	2D	-	DCR	L
OCF3	2D	-	DCR	L
OCF4	2D	-	DCR	L
OCF5	20		DB	
OCF6	43	C	MOV	B, E
OCF7	48	H	MOV	C, B
OCF8	41	A	MOV	B, C
OCF9	4C	L	MOV	C, H
OCFA	4C	L	MOV	C, H
OCFB	45	E	MOV	B, L
OCFC	4E	N	MOV	C, M
OCFD	47	G	MOV	B, A
OCFE	45	E	MOV	B, L
OCFF	52	R	MOV	D, D
OD00	20		DB	
OD01	2D	-	DCR	L
OD02	2D	-	DCR	L
OD03	2D	-	DCR	L
OD04	2D	-	DCR	L
OD05	2D	-	DCR	L
OD06	2D	-	DCR	L
OD07	2D	-	DCR	L
OD08	2B	+	DCX	H
OD09	0D		DCR	C

HT

HT

HT

Address	Hex	Label	Instruction	Op
0D08	0D		DCR	C
0D0B	0D		DCR	C
0D0C	0D		DCR	C
0D0D	34	4	INR	M
0D0E	30	0	DB	
0D0F	0D		DCR	C
0D10	00		NOP	
0D11	00		NOP	
0D12	00		NOP	
0D13	00		NOP	
0D14	00		NOP	
0D15	00		NOP	
0D16	00		NOP	
0D17	00		NOP	
0D18	00		NOP	
0D19	00		NOP	
0D1A	00		NOP	
0D1B	00		NOP	
0D1C	00		NOP	
0D1D	00		NOP	
0D1E	00		NOP	
0D1F	00		NOP	
0D20	72	r	MOV	M,D
0D21	00		NOP	
0D22	01 20 EC		LXI	B,EC20
0D25	09		DAD	B
0D26	8E		ADC	M
0D27	02		STAX	B
0D28	72	r	MOV	M,D
0D29	00		NOP	
0D2A	01 20 EC		LXI	B,EC20
0D2D	09		DAD	B
0D2E	8E		ADC	M
0D2F	02		STAX	B
0D30	72	r	MOV	M,D
0D31	00		NOP	
0D32	01 20 EC		LXI	B,EC20
0D35	09		DAD	B
0D36	8E		ADC	M
0D37	02		STAX	B
0D38	72	r	MOV	M,D
0D39	00		NOP	
0D3A	01 20 EC		LXI	B,EC20
0D3D	09		DAD	B
0D3E	8E		ADC	M
0D3F	02		STAX	B
0D40	72	r	MOV	M,D
0D41	00		NOP	
0D42	01 20 EC		LXI	B,EC20
0D45	09		DAD	B
0D46	8E		ADC	M
0D47	02		STAX	B
0D48	CA 06 CA		JZ	CA06
0D4B	06 A4		MVI	B,A4
0D4D	06 A4		MVI	B,A4
0D4F	06 A4		MVI	B,A4
0D51	06 A4		MVI	B,A4
0D53	06 18		MVI	B,18
0D55	06 73	s	MVI	B,73
0D57	07		RLC	
0D58	CA 06 A4		JZ	A406
0D5B	06 18		MVI	B,18
0D5D	06 73	s	MVI	B,73
0D5F	07		RLC	
0D60	68	h	MOV	L,B
0D61	08		DB	

INPUT Line Buffer

Label ✓

0D63	06 61	#	MVI	B, 61
0D65	C1		POP	B
0D66	61	#	MOV	H, C
0D67	C1		POP	B
0D68	61	#	MOV	H, C
0D69	C1		POP	B
0D6A	B3		ORA	E
0D6B	C0		RNZ	
0D6C	BE		CMP	M
0D6D	C0		RNZ	
0D6E	91		SUB	C
0D6F	C0		RNZ	
0D70	02		STAX	B
0D71	20		DB	
0D72	07		RLC	
0D73	80		ADD	B
0D74	14		INR	D
0D75	0C		INR	C
0D76	CF		RST	1
0D77	09		DAD	B
0D78	00		NOP	
0D79	20		DB	
0D7A	07		RLC	
0D7B	80		ADD	B
0D7C	0A		LDAX	B
0D7D	0D		DCR	C
0D7E	28		DB	
0D7F	03		INX	B
0D80	8D		ADC	L
0D81	00		NOP	
0D82	00		NOP	
0D83	00		NOP	
0D84	00		NOP	
0D85	00		NOP	
0D86	00		NOP	
0D87	00		NOP	
0D88	00		NOP	
0D89	00		NOP	
0D8A	00		NOP	
0D8B	00		NOP	
0D8C	00		NOP	
0D8D	00		NOP	
0D8E	00		NOP	
0D8F	00		NOP	
0D90	00		NOP	
0D91	00		NOP	
0D92	00		NOP	
0D93	00		NOP	
0D94	00		NOP	
0D95	00		NOP	
0D96	00		NOP	
0D97	00		NOP	
0D98	00		NOP	
0D99	00		NOP	
0D9A	00		NOP	
0D9B	00		NOP	
0D9C	00		NOP	
0D9D	00		NOP	
0D9E	00		NOP	
0D9F	00		NOP	
0DA0	00		NOP	
0DA1	00		NOP	
0DA2	00		NOP	
0DA3	00		NOP	
0DA4	00		NOP	

↑ STACK label ✓

0DA6	00			NOP	
0DA7	00			NOP	
0DAB	00			NOP	
0DA9	00			NOP	
0DAA	00			NOP	
0DAB	00			NOP	
0DAC	00			NOP	
0DAD	00			NOP	
0DAE	00			NOP	
0DAF	00			NOP	
0DB0	00			NOP	
0DB1	00			NOP	
0DB2	00			NOP	
0DB3	00			NOP	
0DB4	00			NOP	
0DB5	00			NOP	
0DB6	07			RLC	
0DB7	10			DB	
0DB8	22 43 F4	"C		SHLD	F443
0DBB	09			DAD	B
0DBC	FF			RST	7
0DBD	09			DAD	B
0DBE	00			NOP	
0DBF	04			INR	B
0DC0	21 31 ED	!1		LXI	H,ED31
0DC3	09			DAD	B
0DC4	02			STAX	B
0DC5	0A			LDAX	B
0DC6	0B			DCX	B
0DC7	06 11			MVI	B,11
0DC9	20			DB	
0DCA	F8			RM	
0DCB	09			DAD	B
0DCC	00			NOP	
0DCD	0A			LDAX	B
0DCE	03			INX	B
0DCF	04			INR	B
0DD0	17			RAL	
0DD1	57	W		MOV	D,A
0DD2	F0			RP	
0DD3	09			DAD	B
0DD4	04			INR	B
0DD5	0A			LDAX	B
0DD6	0F			RRC	
0DD7	00			NOP	
0DD8	14			INR	D
0DD9	34	4		INR	M
0DDA	FC 09 EC			CM	EC09
0DDD	09			DAD	B
0DDE	19			DAD	D
0DDF	00			NOP	
0DE0	60	\		MOV	H,B
0DE1	40	@		MOV	B,B
0DE2	06 0A			MVI	B,0A
0DE4	EC 09 DB			CPE	DB09
0DE7	00			NOP	
0DE8	E6 80			ANI	80
0DEA	CA E6 OD			JZ	ODE6
0DED	78	x		MOV	A,B
0DEE	D3 01			OUT	01
0DF0	C9			RET	
0DF1	DB 00			IN	00
0DF3	E6 01			ANI	01
0DF5	C3 F1 OD			JNZ	ODF1
0DF7	DB 01			IN	01

DE6 label ✓

OE3E	00
OE3F	00
OE40	00
OE41	00
OE42	00
OE43	00
OE44	00
OE45	00
OE46	00
OE47	00
OE48	00
OE49	00
OE4A	00
OE4B	00
OE4C	00
OE4D	00
OE4E	00
OE4F	00
OE50	10
OE51	00

[illegible]